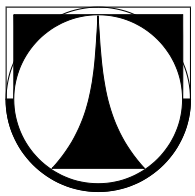


TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

DIPLOMOVÁ PRÁCE

Metody potlačení nežádoucího rušení v telefonním řečovém signálu

Methods of suppressing the ineligible disturbance in telephone speech signal

Liberec 2003 **Jan Kos**

Anotace

Diplomová práce (dále jen DP) se zabývá především dvěma typy rušení, se kterými se často lze setkat v souvislosti s telefonními (obecně akustickými) přístroji – šumem a ozvěnou. Na obecné úrovni je zde probrána problematika jejich vzniku a nežádoucího projevu. Dále se DP velmi detailně věnuje metodám potlačení těchto rušení. V případě ozvěny jde o aplikaci adaptivních filtrů velmi vysokých řádů. Popsány jsou základní adaptivní algoritmy LMS (Least Mean Square) a NLMS (Normalized LMS), hlavně pak pokročilý AP (Affine Projection) algoritmus. Rovněž je diskutována problematika řízení těchto algoritmů. V případě šumu se DP zabývá metodami založenými na spektrální modifikaci signálu, které vycházejí ze „základní“ Wienerovy filtrace. Vyústěním DP je pak funkční software pro potlačení šumu a ozvěny, založený na popsaných metodách. Diskutován a testován je také přínos softwaru pro počítačové zpracování řeči, kterému je rovněž část DP věnována.

Annotation

The diploma thesis (”DT”) deals with noise and echo as the two main types of disturbances usually encountered in acoustic (telephone, in particular) communication. The creation and ineligible effects of the disturbances are discussed in general terms and, further, the disturbance suppression methods are studied in detail. In the case of echo this means applications of very high-order adaptive filters and the DT describes basic adaptive LMS (Least Mean Square) and NLMS (Normalized LMS), and principally, the advanced AP (Affine Projection) algorithms. The problematic of their respective control is also discussed. In the case of noise the DT focuses on the speech signal spectral modification methods derived from the standard Wiener filtration. Functional software for suppressing noise and echo based on the above methods is the main result of the DT. Last part of the DT describes basic methods of computer speech recognition, for which the contribution of the developed software is also discussed and tested.

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé DP a prohlašuji, že **souhlasím** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

Obsah

1 Úvod

Jak již naznačuje název, byla v této práci hlavním předmětem mého zájmu problematika rušení telefonního řečového signálu. Jelikož jde o problematiku poměrně rozsáhlou a komplikovanou, bylo nutné se při jejím zkoumání omezit na určitou třídu problémů a z toho plynoucí i řešitelnost těchto problémů. To je ale celkem běžný postup při řešení většiny technických úloh a proto jej není třeba vnímat jako nějaké omezení, nýbrž jako ověřený postup vedoucí k dobrým výsledkům. Záměrně se zde vyhýbám detailům, neboť ty budou předmětem dalších několika kapitol.

V této úvodní části mi naopak připadá vhodné zmínit „poslání“ diplomové práce. Ta se především skládá ze dvou částí – dokumentační a technické či realizační.

Dokumentační část popisuje problematiku rušení, od základů přes popis a ukázky konkrétních metod dané rušení potlačujících. Zabývá se rovněž využitím těchto metod v některých aplikacích. Jde o nosnou část diplomové práce a důraz je kladen především na dva typy rušení – šum a ozvěnu. Problematika vzniku a hlavně potlačení těchto rušení je zde probrána dosti podrobně. Náplň textu tvoří jak poznatky načerpané z cizí literatury (kterou samozřejmě vždy cituji), tak i výsledky mého vlastního bádání. Dále se práce věnuje automatickému zpracování a rozpoznávání řeči pomocí počítače, jakožto navazující aplikaci, ve které je jakékoliv rušení velmi nežádoucí. Tato problematika je však ve srovnání s rušením probrána o něco méně podrobně, neboť ji přeci jen považuji, co se zaměření této práce týče, spíše za okrajovou. Celá tato dokumentační část má spíše učebnicový charakter, což, jak doufám, je její výhodou. Měla by tak případnému čtenáři poskytnout dostatečné množství informací pro pochopení dané problematiky.

Naproti tomu technická část je softwarovou realizací zde popsaných principů a hlavním cílem diplomové práce. Vznikla pochopitelně dříve než tato část dokumentační a mou snahou v ní bylo hlavně ověřit metody a realizovat software pro potlačení rušení. Jeho výsledky budou součástí následujících stránek a samotný software jako celek bude popsán ke konci této práce v samostatné kapitole. Samozřejmě nebude chybět ani jednoduchý program pro rozpoznávání řeči.

Na tomto místě bych ještě rád poděkoval panu Prof. Janu Nouzovi za neocenitelné informace týkající se zpracování řeči a panu Ericu J. Diethornovi za pomoc při zkoumání problematiky šumu.

Dále musím upozornit na to, že téměř celá práce, kromě této úvodní a závěrečné kapitoly, je napsána v tzv. my-formě. Důvodem samozřejmě není větší počet autorů, byl jsem pouze toho názoru, že práce tak bude lépe čitelná.

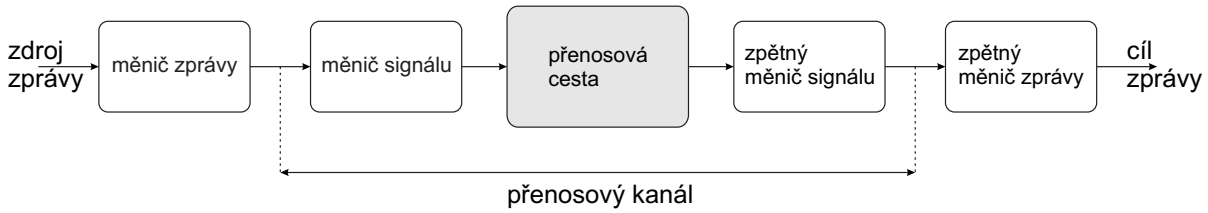
Nyní se již budu zabývat problematikou rušení a dalšími věcmi, které s ní souvisí...

2 Seznámení s problematikou

Začněme u běžného telefonu. V současnosti je možné vést telefonní hovor z prakticky libovolného místa na Zemi a život bez tohoto přístroje si dnes lze jen těžko představit. Dnes je celkem běžné, že se hovor na své cestě od jednoho mluvčího k druhému šíří velmi různorodými způsoby. Část své cesty urazí např. pevnou telefonní sítí, část sítí mobilní a na větší vzdálenosti připadá v úvahu i komunikace družicová. To nás ovšem, jako řešitele nějakého technického problému, staví do poměrně nepříjemné situace, kdy naprosto nejsme předem schopni cokoliv říci o povaze této přenosové cesty či kanálu. S tím se bohužel musíme smířit, ačkoliv pro uvažované aplikace není tento problém nijak zásadní. Většinou totiž budeme uvažovat hovory místní a navíc rušení, kterým se zde budeme zabývat, je obecného charakteru a není proto třeba si situaci komplikovat uvažováním mnoha různých přenosových cest. V této práci proto nepůjdeme příliš hluboko do problematiky telekomunikační techniky a případné zájemce tímto odkazujeme na knihy [4] a [3], kde lze nalézt mnoho zajímavých informací a ze kterých jsme rovněž čerpali.

Telefonní kanál

Zjednodušenou situaci jednosměrného telefonního hovoru ukazuje obrázek (1). Promluva jednoho mluvčího je mikrofonem převedena na elektrický signál. Následuje jeden nebo více přenosových kanálů, z nichž každý je složen ze tří nejdůležitějších částí. Tou první je měnič signálu, který přizpůsobí elektrický signál odpovídající promluvě mluvčího dané přenosové cestě. Následuje přenosová cesta, o které ještě budeme mluvit. Třetí částí přenosového kanálu je zpětný měnič signálu, jehož význam je zřejmý. Za přenosovým kanálem následuje ještě zpětný měnič zprávy, kterým je v běžném hovoru samozřejmě reproduktor. K tomuto obrázku se později ještě vrátíme.



Obrázek 1: Schéma hovoru jedním směrem

To jsme ovšem uvažovali pouze jednosměrný přenos (hovor). K obousměrnému přenosu potřebujeme ještě jeden podobný kanál pro směr opačný, čímž vytvoříme tzv. *telekomunikační okruh*, tedy dvojici vzájemných protisměrných kanálů umožňujících obousměrnou komunikaci. Máme-li k dispozici tento okruh, může komunikace probíhat dvěma základními způsoby nazývanými *simplex* (*poloduplex*) a *duplex*. Při simplexu probíhá komunikace okruhem střídavě v jednom či druhém směru. Při duplexu naopak probíhá komunikace současně v obou směrech.

Zastavme se nyní u samotné přenosové cesty. Ta může být realizována různými způsoby, z nichž nejběžnější je metalické vedení. V tomto případě je rušení poměrně častým jevem, nicméně vhodným uspořádáním vodičů (symetrické či koaxiální vedení) lze takové rušení výrazně potlačit. Jinou realizací přenosové cesty je dnes běžné optické vlákno, které velmi účinně eliminuje pronikání rušení (okolního světla) do svého jádra a je z tohoto hlediska tudíž velmi odolným přenosovým médiem. Do třetice ještě zmiňme rádiové vlny šířící se volným prostorem. To bylo v krátkosti o přenosové cestě, nyní se pokusíme něco říci o samotném telefonním kanálu.

Z uvedeného je už asi zřejmé, že to co nazveme telefonním kanálem, může být v dnešní době ve skutečnosti celá řada kanálů navazujících a je proto velmi těžké tuto soustavu nějak popsat. Základní charakteristikou telefonního kanálu je přenášené frekvenční pásmo, běžně se udává rozsah 300 Hz až 3400 Hz. Šířka pásma je tedy 3100 Hz. Jde pouze o informativní a běžně akceptované údaje platné jak pro klasické pevné, tak i mobilní telefonní sítě. Skutečnost může být trochu jiná.

Rušení

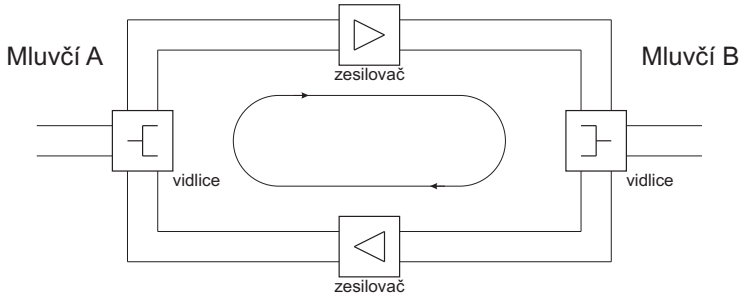
Možná vám připadá, že chodíme stále kolem horké kaše, neboť problematiky rušení jsme se zatím dotkli pouze lehce. Pokusíme se to napravit. Tím nejběžnějším rušením, se kterým se jistě setkal každý, je šum. Člověk je poměrně shovívavý a nepřekročí-li šum určitou mez, dokáže se s ním dobře vyrovnat. Situace ale nemusí být únosná, probíhá-li komunikace nikoliv mezi dvěma lidmi, nýbrž mezi člověkem a počítačem. Právě kvůli tomuto druhému případu se budeme šumu v této práci věnovat i my. Vše podstatné bude probráno v kapitole 4 a proto přejděme k dalšímu typu rušení.

Tímto dalším rušením je tzv. ozvěna, se kterou se také jistě setkal každý. Uvažujme situaci, kdy vedeme telefonní hovor a pokaždé, když něco řekneme, slyšíme znovu sami sebe v telefonním sluchátku. Jde o klasickou ozvěnu, která mnohdy působí mnohem rušivěji nežli šum. Zvláště markantní je tento efekt při používání hands-free sad, např. v automobilu. A jak vlastně ozvěna vzniká? Velmi jednoduše...

Všechno co řekneme do mikrofonu je přeneseno k druhému mluvčímu, který nás slyší díky svému reproduktoru. Zachytává-li ovšem jeho mikrofon i tento náš reprodukovaný hlas, slyšíme se logicky dvakrát. Na vině tak není náš telefonní přístroj, nýbrž přístroj druhého mluvčího. V tomto případě se jedná o ozvěnu *akustickou*. Systémy pro potlačení této ozvěny bývají často součástí vlastních přístrojů.

Zamyslíme-li se nyní nad uvedenými rušeními, patrně zjistíme, že mohou mít jednu společnou příčinu. Tou příčinou je první člen řetězce z obrázku (1), kterým je mikrofon. Je to právě mikrofon, který může během hovoru zachytávat okolní nežádoucí šum či hluk nebo reprodukovaný hlas mluvčího. Podobným nežádoucím způsobem může ovšem působit i reproduktor a proto vlastně nejvíce záleží na vzájemném uspořádání těchto dvou prvků. Jejich uspořádání by mělo být takové, aby se vzájemně „neviděly“. V současnosti lze tento problém dobře řešit použitím směrových mikrofonů či reproduktorů, i když i toto řešení má svá úskalí. Těžko si asi představíme směrový mikrofon např. u hands-free sady. V podobných případech je třeba přistoupit k jiným řešením.

Vraťme se nyní ještě k ozvěně. Kromě akustické ozvěny se ještě můžeme setkat s ozvěnou *elektrickou*, která sice působí stejně, vzniká ovšem jiným způsobem. K vysvětlení jejího vzniku nám pomůže obrázek (2).



Obrázek 2: Vznik elektrické ozvěny

Obrázek znázorňuje část telefonní sítě. Znázorňuje místo, ve kterém se z dvouvodičového vedení přechází na vedení čtyřvodičové a naopak. To se děje z důvodu zesílení hovorového signálu, který je třeba přenést na velkou vzdálenost. I když je totiž možné přenášet po jednom vedení obousměrnou komunikaci, není možné vyrobit zesilovač zesilující v obou směrech. Z toho důvodu se přechází na vedení čtyřvodičové, kde již zesílení nepůsobí problémy. Prvek, který takový přechod zajistí, se nazývá *telekomunikační vidlice* a je to právě ona, kdo stojí za vznikem ozvěny.

Představme si promluvu mluvčího A. Signál se šíří směrem k mluvčímu B a telekomunikační vidlice jej „přesměruje“ do horní větve. V té je signál zesílen a poté je druhou vidlicí „poslán“ k mluvčímu B. Problémem je, že vidlice není ideální a část tohoto signálu propustí i do spodní větve. Ve spodní větvi je signál opět zesílen a první vidlicí „poslán“ zpět do místa svého vzniku. Tím vznikne ozvěna. Ta se samozřejmě liší podle délky vedení a kvality vidlic. Není ovšem nijak zaručeno, že se tentýž proces nebude opakovat i pro nově vzniklou ozvěnu, což může vést ke vzniku ozvěny několikanásobné.

Tolik tedy na úvod k problematice rušení. V následujících kapitolách se popsaným věcem budeme věnovat mnohem podrobněji. Ještě dodejme něco k zajímavé literatuře. Jako vynikající zdroj informací týkající se problematiky ozvěny nelze než doporučit články [9] a [2], ze kterých jsme vydatně čerpali. Co se šumu týče, pozornost patří především výborné knize [1].

Barge-In

Zkusíme si nyní vysvětlit ještě jeden důležitý pojem – barge-in. Začneme trochu zeširoka. Již jsme se lehce zmínili o dialogovém systému, který hovoří s člověkem. Takový systém pracuje v současné době nejčastěji na poměrně jednoduchém principu. K systému je možné se připojit např. pomocí telefonu. Po připojení nám „inteligence“ systému začne klást otázky podle svého zaměření a samozřejmě očekává příslušné odpovědi. Hlavní část takového systému – rozpoznávač řeči – naše odpovědi vyhodnocuje a předává vyšší vrstvě, která řídí vlastní dialog.

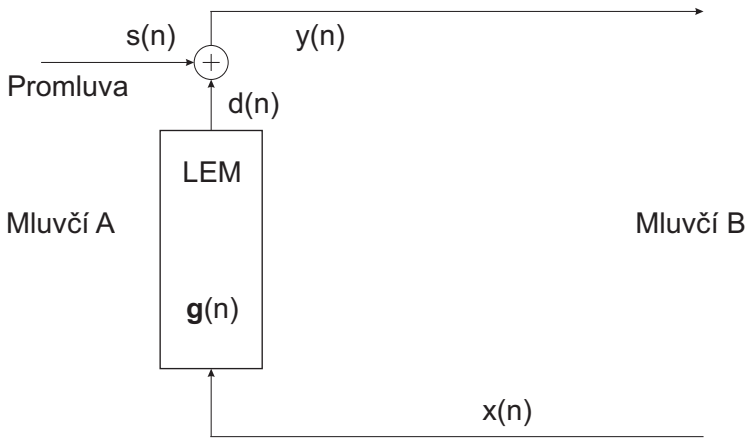
Velkým problémem je právě rušení, které je příčinou špatného rozpoznávání. Uvažujme nyní ozvěnu, která proniká do dialogového systému. Systém k nám promlouvá a tato promluva (ovlivněná signálovou cestou) se vrací zpět na jeho vstup. V takovém případě jsou samozřejmě výsledky rozpoznávače nesmyslné. Tento problém se v současné době řeší tak, že po dobu promluvy systému je rozpoznávač neaktivní. Zdá se to být logické řešení, neboť během promluvy systému posluchač naslouchá a odpoví až po dokončení otázky. Toto řešení má ovšem úskalí – otázky či nabídky systému bývají poměrně dlouhé a člověk si je po několika připojeních snadno zapamatuje. Je potom velmi nepříjemné stále čekat na jejich dokončení.

Snahou tedy je umožnit člověku tzv. *nabourání se* do otázky systému. Tento přístup je označován jako barge-in. Není potom nutné čekat na dokončení otázky, protože systém naše nabourání rozpozná a pohotově zareaguje. Z uvedeného tedy vyplývá, že dialogový systém typu barge-in musí obsahovat podsystém potlačující ozvěnu. Jen tak může být rozpoznávač aktivní i během promluvy k člověku, neboť ozvěna se na vstup systému vůbec nedostane.

3 Ozvěna a její potlačení

3.1 Úvod

Základní informace jsme si již vysvětlili v předchozí kapitole. Podívejme se nyní na obrázek (3), který znázorňuje situaci při běžném telefonním hovoru (přesněji polovinu situace pro jednoho mluvčího). Budeme zatím uvažovat ozvěnu akustickou.



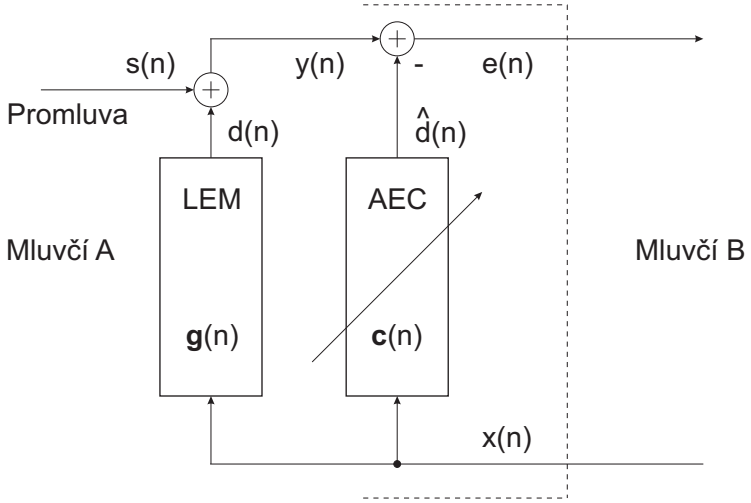
Obrázek 3: Jednoduché schéma komunikačního přístroje

Nejdůležitějším prvkem na obrázku je tzv. LEM (z anglického loudspeaker-enclosure-microphone) systém odpovídající přenosové cestě akustického signálu u přístroje mluvčího A. Tento systém si můžeme jednoduše představit jako soustavu reproduktor-mikrofon. Signál vstupující do tohoto systému je tedy reprodukován a po určité době odpovídající jeho šíření směrem k mikrofonu je tímto mikrofonem také zachycen. Každý takovýto LEM systém je charakterizován svou impulsní odezvou $\mathbf{g}(n)$ (k detailům se dostaneme později).

Promluva přicházející od mluvčího B tedy prochází tímto systémem, který ji určitým způsobem pozmění a k takto vzniklé *ozvěně* se ještě přičítá promluva (signál $s(n)$) mluvčího A. Možná je trochu překvapující, že promluva mluvčího A je zakreslena mimo LEM systém. To je nicméně v pořádku, neboť LEM systém charakterizuje pouze přenosovou cestu mezi reproduktorem a mikrofonem a promluva mluvčího s ním nesouvisí. Signál $s(n)$ tak v sobě zahrnuje i vlastnosti mikrofonu a vše, co tento mikrofon zachytává, např. i šum. Vidíme tedy, že signál $y(n)$ přicházející k mluvčímu B obsahuje kromě užitečné části $s(n)$ i nežádoucí ozvěnu $d(n)$.

Řešení tohoto problému je, alespoň v principu, velmi prosté. K vlastnímu LEM systému připojíme paralelně další systém, který bude jeho modelem a který zajistí potlačení ozvěny. Obrázek (4) znázorňuje celou situaci. Oba systémy jsou nyní součástí daného (telefonního) přístroje.

Řešení je asi již zřejmé, přesto jej ale vysvětlíme. Je-li druhý systém, tzv. AEC (z angl. acoustic echo canceller) dobrým modelem systému LEM, „produkují“ oba dva stejný výstup a ozvěna je tak ze signálu $y(n)$ odečtena. K mluvčímu B se tudíž dostane pouze užitečný signál $s(n)$. Realizace tohoto jednoduchého principu je vskutku nesnadná. Je totiž samozřejmě třeba zajistit, aby systém AEC byl opravdu *přesným* modelem systému LEM, což je největším



Obrázek 4: Princip systému pro potlačení ozvěny

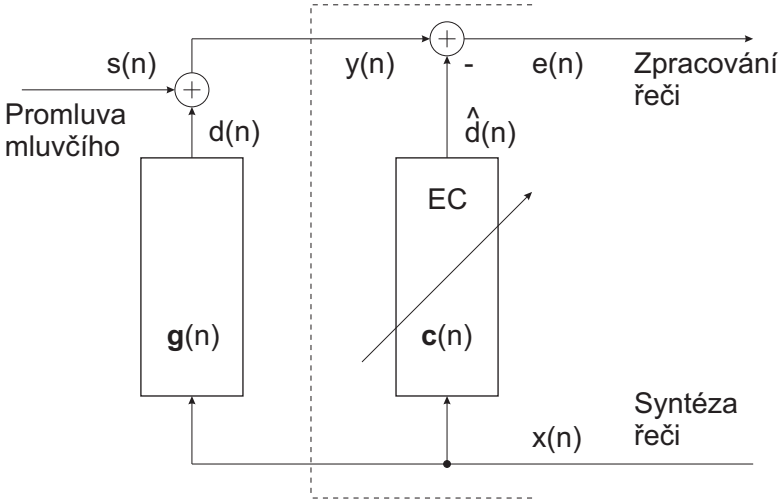
problémem celého zpracování. Tím prvním řešením, které nás zřejmě napadne, je změření impulsní odezvy systému LEM a následná technická realizace této odezvy systémem AEC. To ale bohužel není možné, je třeba si totiž uvědomit, že skutečný systém LEM se během hovoru neustále mění. Stačí např. mírná změna teploty nebo se vůči přístroji mírně pohnout (přitisknout k uchu) a impulsní odezva systému se okamžitě změní. Z tohoto důvodu je třeba, aby systém AEC byl *adaptivní*, tj. neustále (a co nejrychleji) se během hovoru přizpůsoboval svému vzoru.

Konkrétní postupy, jak toho docílit, si vysvětlíme později. Nyní se zamysleme nad principem. Abychom mohli nějakým způsobem adaptovat model na skutečný systém, musíme znát míru jejich shody. Uvažujme nyní situaci, kdy je signál $s(n)$ nulový (A nemluví). Rovněž uvažujme, že signál $x(n)$ naopak nulový není, neboť pak by systémy nebyly ničím buzeny a nebylo by možné je žádným způsobem porovnávat. Za tohoto předpokladu nám o podobnosti systémů dobře vypovídá rozdílový signál $e(n)$, který je pro ideální model nulový. Na minimalizaci tohoto signálu je potom možné adaptaci založit. Situace se dosti podstatně komplikuje, není-li signál $s(n)$ nulový. V tomto případě nemůže adaptace modelu vůbec probíhat, neboť signál $e(n)$ již o shodě nijak nevypovídá. Adaptaci je tak třeba zastavit a „spoléhat“ na to, že se systém LEM nebude příliš měnit. Tuto důležitou situaci, kdy jsou „aktivní“ oba mluvčí, nazýváme *dvojí řeč* (z angl. *double-talk*). Je třeba zdůraznit, že není nijak snadné tuto situaci rozpoznat (při zpracování máme k dispozici pouze signály $x(n)$ a $y(n)$, nikoliv $d(n)$ či $s(n)$). Jde ovšem o nesmírně důležitou část systému a správný algoritmus rozpoznávající dvojí řeč je naprosto klíčový. Konkrétní metodě založené na korelaci signálů $x(n)$ a $y(n)$ se budeme věnovat ve zvláštní kapitole.

Tím jsme si vysvětlili nejdůležitější principy systému pro potlačení ozvěny. Uvažovali jsme ovšem ozvěnu akustickou a systém pro její potlačení byl součástí samotného (telefonního) přístroje. Zdá se tedy, že uvedené informace jsou vhodné pouze pro výrobce daných přístrojů. Naštěstí tomu tak není a všechny systémy potlačující ozvěnu jsou si velmi podobné. Situace by se proto příliš nezměnila, pokud bychom např. v telefonní ústředně potlačovali ozvěnu elektrickou. Původcem ozvěny by již nebyl LEM systém, ale rozhodně systém s podobnými

vlastnostmi.

Nás ovšem převážně zajímá ještě jiná situace. Uvažujeme případ, kdy spolu komunikují člověk a počítač. Je třeba, aby dialogový systém, komunikující s člověkem, zajistil potlačení ozvěny, o jejímž původu ovšem nelze předem nic říci. Může být elektrická či akustická a pokaždé navíc velmi rozdílná, neboť k dialogovému systému se může po telefonu připojit téměř kdokoliv a odkudkoliv. Každý hovor tak probíhá z jiného místa i přístroje. Systém pro potlačení ozvěny (nyní pouze EC) je proto třeba přenést do vlastního dialogového systému, jak ukazuje obrázek (5). Jak ale vidíme, změna oproti předchozímu obrázku je nepatrná a způsob řešení se rovněž nemění. Jen je dobré si uvědomit, že systém $\mathbf{g}(n)$ nyní představuje kompletní přenosovou cestu od počítače k mluvčímu a zpět.



Obrázek 5: Rozšíření dialogového systému o potlačení ozvěny

3.2 Adaptivní algoritmy

Úvodní seznámení

V této části naší práce se budeme věnovat adaptivním algoritmům – od základních principů až po ukázky konkrétních výpočetních postupů. Mnoho již bylo naznačeno v úvodní části celé této kapitoly, přesto budeme postupovat od začátku. Vraťme se proto k obrázkům (4) a (5), které celou situaci ilustrují nejlépe. Prozatím budeme předpokládat, že signál $s(n)$ je nulový, neboť tento signál „nepatří“ do naší adaptační úlohy. Brzy uvidíme proč.

Na obrázcích tedy máme vedle sebe zapojeny dva systémy. Jeden z nich je systém, jehož parametry neznáme a snažíme se je zjistit. Druhý systém představuje náš model. Jde vlastně o běžnou identifikační úlohu. Otázkou zůstává, jakým způsobem budeme systémy popisovat. Velmi efektivním a zdaleka nejpoužívanějším způsobem je popis těchto systémů pomocí struktury FIR (Finite Impulse Response – konečná impulsní odezva). Popsat tyto systémy lze samozřejmě i jinými způsoby, nicméně FIR má některé zásadní výhody. Tou největší je zřejmě zaručená stabilita takto popsaného systému. Jinou velkou výhodou je jednoduchost matematického popisu a s tím souvisejících výpočtů.

Každý z našich dvou systémů je popsán svou unikátní impulsní odezvou. Skutečný systém, odpovídající přenosové cestě signálu (elektrického či akustického), je dán impulsní odezvou $\mathbf{g}$, přesněji řečeno vektorem $\mathbf{g}$, který představuje hodnoty této impulsní odezvy. Je dobré si uvědomit, že tato odezva je teoreticky nekonečně dlouhá (což je vlastně v rozporu se strukturou FIR) a výstup tohoto systému je proto dán vztahem (uvažujeme $s(n) = 0$)

$$y(n) = d(n) = \sum_{i=0}^{\infty} g_i x(n-i) \tag{3.1}$$

Naproti tomu druhý systém představuje model systému prvního a pro potřeby výpočtu musíme jeho impulsní odezvu $\mathbf{c}$ uvažovat samozřejmě konečnou. Je-li tedy M počet hodnot impulsní odezvy, je výstup dán vztahem

$$\hat{y}(n) = \hat{d}(n) = \sum_{i=0}^{M-1} c_i x(n-i) \tag{3.2}$$

Úloha adaptace spočívá v nalezení co nejpřesnějšího modelu $\mathbf{c}$ systému $\mathbf{g}$. Je-li totiž systém $\mathbf{c}$ dostatečně přesným modelem, jsou výstupy obou systémů shodné (mají stejné buzení $x(n)$) a chybový signál $e(n)$ je nulový. A to je přesně náš cíl.

Velmi důležitou je volba počtu koeficientů M modelu $\mathbf{c}$. Tato volba je samozřejmě silně závislá na úloze, nicméně uvažujeme-li systém pro potlačení ozvěny, je třeba volit M opravdu dosti velké. To je dáno dlouhou impulsní odezvou skutečného systému, kterou je třeba dobře modelovat. V současnosti se běžně užívají filtry o délce až 1024 koeficientů, vystačit lze i s hodnotami menšími, většinou ale na úkor kvality systému. Pro menší hodnoty naopak hovoří časová nenáročnost výpočtů, která může být důležitým požadavkem.

Dříve než se budeme zabývat vlastními adaptivními algoritmy, zavedeme si některá dále používaná značení. Označme

$\mathbf{c}_M(n)$ vektor $(c_0(n), c_1(n), \ldots, c_{M-1}(n))^{\mathrm{T}}$ o délce M,

$\mathbf{X}_{M,L}(n)$ matici rozměru $M \times L$ a speciálně

$\mathbf{x}_M(n)$ vektor sestávající z M posledních hodnot buzení $x(n)$, tedy
$(x(n), x(n-1), x(n-2), \ldots, x(n-M+1))^{\mathrm{T}}$

Výstup modelu ze vztahu (3.2) může být potom zapsán jako

$$\hat{y}(n) = \mathbf{x}_M^{\mathrm{T}}(n)\mathbf{c}_M(n) = \mathbf{c}_M^{\mathrm{T}}(n)\mathbf{x}_M(n). \tag{3.3}$$

Princip algoritmů

Naším úkolem je tedy nalézt co nejlepší impulzní odezvu filtru danou vektorem koeficientů $\mathbf{c}_M$. Není samozřejmě v našich silách určit tuto odezvu okamžitě, ale je třeba se ke skutečné odezvě co nejrychleji přibližovat. Je asi jasné, že s každým příchodem nových vzorků $x(n)$ a $y(n)$ se

naše informace o chování systému rozšíří a my tak můžeme aktualizovat i náš model. Základní formulka pro výpočet odezvy filtru má tedy následující rekurzivní charakter

$$\mathbf{c}_M(n) = \mathbf{c}_M(n-1) + \mu(n)\mathbf{v}_M(n). \tag{3.4}$$

Formulka je v principu velmi jednoduchá. V každém výpočetním kroku se určí vektor $\mathbf{v}_M$, který vlastně udává směr změny parametrů filtru $\mathbf{c}_M$. Způsobů, jak určit tento směr, existuje nesmírné množství, vychází se z nejrůznějších kritérií a jde právě o klíčovou věc každého algoritmu. K detailům se dostaneme později. Parametr μ (skalár), tzv. *krok algoritmu*, reguluje vliv vektoru $\mathbf{v}_M$ na změnu koeficientů filtru $\mathbf{c}_M$ a jeho správné určení v každém výpočetním kroku je zásadní pro jakýkoliv systém pro potlačení ozvěny. Vidíme tedy, že máme-li systém dobře navržen, můžeme se s modelem tímto způsobem po několika krocích přiblížit skutečné odezvě.

Pojďme se nyní zabývat volbou směrového vektoru $\mathbf{v}_M$. Nejčastější způsob jeho určení vychází z nějaké zatím blíže nespecifikované kriteriální funkce, která je závislá na vektoru filtru $\mathbf{c}_M$. Označme si tuto funkci $V(\mathbf{c}_M)$. Je žádoucí, aby funkce $V(\mathbf{c}_M)$ dobře vypovídala o shodě mezi skutečným systémem a modelem a aby při úplné shodě nabývala svého minima. Víme totiž, že obrácený gradient udává směr největšího spádu a je-li naše funkce navržena popsaným způsobem, je potom nejlepší postupovat při aktualizaci koeficientů filtru $\mathbf{c}_M$ v tomto směru. Označíme-li gradient jako $\mathbf{g}_M$, volíme potom směrový vektor $\mathbf{v}_M$ jako

$$\mathbf{v}_M(n) = -\mathbf{g}_M(n). \tag{3.5}$$

Vztah (3.4) lze považovat za výchozí. Je sice naprosto dostačující, vetšinou se ale využívá jeho rozšířená varianta, která zdánlivě nabízí více možností a je rovněž o něco přehlednější. Její předpis je následující

$$\mathbf{c}_M(n) = \mathbf{c}_M(n-1) - \frac{1}{2}\mu(n)\mathbf{W}_M(n)\mathbf{g}_M(n), \tag{3.6}$$

kde $\mathbf{W}_M$ je tzv. váhová matice. U jednodušších algoritmů se uvažuje jednotková.

Jednou z neužívanějších kriteriálních funkcí bývá střední kvadratická odchylka mezi skutečným a žádaným výstupem podle vztahu (E je operátor střední hodnoty)

$$V(\mathbf{c}_M) = \mathrm{E}[e(n)^2] = \mathrm{E}[(y(n) - \hat{y}(n))^2]. \tag{3.7}$$

Tato funkce má pro nás hlavně teoretický význam, neboť většinou není možné střední hodnotu určit a operátor E je třeba nahradit něčím více „praktickým“. I tak si ale ukažme, jakým způsobem lze určit gradient této funkce. Jednoduchou úpravou dostaneme

$$V(\mathbf{c}_M) = \mathrm{E}[y(n)^2 - 2y(n)\hat{y}(n) + \hat{y}(n)^2]. \tag{3.8}$$

Abychom si vztahy trochu zjednodušili, budeme vynechávat agrument, který je všude n, pouze u vektoru $\mathbf{c}_M$ má být $n-1$, neboť v n-tém výpočetním kroku známe pouze vektor $\mathbf{c}_M(n-1)$ a vektor $\mathbf{c}_M(n)$ získáme až ke konci výpočtu podle vztahu (3.6).

Dosazením do (3.8) dle definice dostaneme vztah

$$V(\mathbf{c}_M) = \mathrm{E}[y^2 - 2y\mathbf{x}_M^{\mathrm{T}}\mathbf{c}_M + \mathbf{c}_M^{\mathrm{T}}\mathbf{x}_M\mathbf{x}_M^{\mathrm{T}}\mathbf{c}_M], \tag{3.9}$$

který můžeme ještě upravit do tvaru

$$V(\mathbf{c}_M) = \mathrm{E}[y^2] - 2\mathbf{d}_M^{\mathrm{T}}\mathbf{c}_M + \mathbf{c}_M^{\mathrm{T}}\mathbf{R}_M\mathbf{c}_M, \tag{3.10}$$

kde $\mathbf{R}_M$ je autokorelační matice vstupního signálu $x(n)$ a $\mathbf{d}_M$ je vzájemný korelační vektor mezi vstupním signálem $x(n)$ a požadovaným výstupním signálem $y(n)$, matematicky zapsáno

$$\mathbf{R}_M = \mathbf{R}_M^{\mathrm{T}} = \mathrm{E}[\mathbf{x}_M\mathbf{x}_M^{\mathrm{T}}], \quad \mathbf{d}_M = \mathrm{E}[\mathbf{x}_M y]. \tag{3.11}$$

Gradient funkce $V(\mathbf{c}_M)$ dostaneme derivováním[1] podle parametru $\mathbf{c}_M$.

$$\mathbf{g}_M = -2\mathbf{d}_M + (\mathbf{R}_M + \mathbf{R}_M^{\mathrm{T}})\mathbf{c}_M = -2\mathbf{d}_M + 2\mathbf{R}_M\mathbf{c}_M. \tag{3.12}$$

Výsledek je to sice zajímavý, nicméně, jak jsme již upozornili, hlavně teoretický. Opět totiž není dost dobře možné určovat během výpočtu onu autokorelační matici a vzájemný korelační vektor. Z těchto důvodů se běžně používají jiné varianty funkce $V(\mathbf{c}_M)$ než (3.7). Pojďme se nyní posunout od teorie k praxi a ukázat si některé konkrétní adaptační algoritmy.

Algoritmy LMS a NLMS

Tyto dva algoritmy jsou z adaptivních patrně nejznámější a patří k těm jednodušším. Zkratka LMS je z anglického Least Mean Squares, NLMS k tomuto názvu ještě přidává Normalized. Oba algoritmy spolu s mnoha jejich obdobami patří do skupiny gradientních algoritmů, jejichž společným rysem je jednotková váhová matice ze vztahu (3.6), tedy $\mathbf{W}_M{=}\mathbf{I}_M$, což vede na

$$\mathbf{c}_M(n) = \mathbf{c}_M(n-1) - \frac{1}{2}\mu(n)\mathbf{g}_M(n). \tag{3.13}$$

K odvození obou algoritmů je dále potřeba nějakým způsobem aproximovat operátor střední hodnoty ve vztahu (3.7). Nejjednodušším způsobem je jeho prosté vynechání, čímž se funkce $V(\mathbf{c}_M)$ stane závislou pouze na datech v aktuálním kroku, tj.

$$V(\mathbf{c}_M) = e(n)^2 = (y(n) - \hat{y}(n))^2, \tag{3.14}$$

což opět po dosazení vede na

$$V(\mathbf{c}_M) = (y(n) - \mathbf{x}_M^{\mathrm{T}}(n)\mathbf{c}_M(n-1))^2. \tag{3.15}$$

Nás ale zajímá především gradient této funkce, který dostaneme derivováním pole $\mathbf{c}_M$, tj.

$$\mathbf{g}_M(n) = -2\big[y(n) - \mathbf{x}_M^{\mathrm{T}}(n)\mathbf{c}_M(n-1)\big]\mathbf{x}_M(n), \tag{3.16}$$

což lze jednoduše upravit do tvaru

$$\mathbf{g}_M(n) = -2\mathbf{x}_M(n)e(n). \tag{3.17}$$

[1] platí následující vztahy (derivace skaláru podle vektoru $\mathbf{x}$)

$$(\mathbf{a}^{\mathrm{T}}\mathbf{x})' = \mathbf{a}, \quad (\mathbf{x}^{\mathrm{T}}\mathbf{b})' = \mathbf{b}, \quad (\mathbf{x}^{\mathrm{T}}\mathbf{C}\mathbf{x})' = (\mathbf{C} + \mathbf{C}^{\mathrm{T}})\mathbf{x}$$

Zbývá již pouze dosadit tento gradient do (3.13), čímž bude odvození algoritmu LMS hotové, tedy

$$\mathbf{c}_M(n) = \mathbf{c}_M(n-1) + \mu(n)\mathbf{x}_M(n)e(n). \tag{3.18}$$

V této podobě již můžeme algoritmus začít okamžitě využívat, jednotlivé členy lze v každém kroku bez problémů vypočítat. Otázkou nicméně stále zůstává volba koeficientu μ, kterou se ovšem budeme detailněji zabývat později pouze pro algoritmus AP. Případný zájemce nechť experimentuje.

Pojďme se nyní posunout k algoritmu NLMS. Ten vznikne z předchozího jednoduchou normalizací druhého členu (3.18) koeficientem

$$\frac{1}{\mathbf{x}_M^{\mathrm{T}}(n)\mathbf{x}_M(n)}, \tag{3.19}$$

což vede k výslednému vztahu

$$\mathbf{c}_M(n) = \mathbf{c}_M(n-1) + \mu(n)\frac{\mathbf{x}_M(n)}{\mathbf{x}_M^{\mathrm{T}}(n)\mathbf{x}_M(n)}e(n). \tag{3.20}$$

Aby se předešlo případným problémům při dělení, tento vztah se ještě lehce upravuje do tvaru

$$\mathbf{c}_M(n) = \mathbf{c}_M(n-1) + \mu(n)\frac{\mathbf{x}_M(n)}{\alpha + \mathbf{x}_M^{\mathrm{T}}(n)\mathbf{x}_M(n)}e(n), \tag{3.21}$$

kde α je nějaké malé číslo $\alpha > 0$. Pro tento algoritmus doporučují autoři [9] volit parametr $\mu(n)$ v intervalu $(0, 2)$.

Jednoduchým rozšířením algoritmu LMS vznikl algoritmus, který má podstatně lepší vlastnosti a který je navíc díky své jednoduchosti široce využíván v běžných aplikacích.

Algoritmus AP

Předchozí algoritmy byly poměrně jednoduché a lze tudíž očekávat, že v náročnějších aplikacích nebudou jejich vlastnosti dostatečné. Proto se nyní budeme zabývat dalším algoritmem, jehož zkratka AP je z anglického Affine Projection. Dlužno říci, že tento algoritmus již zdaleka není jednoduchý a stejně tak není jednoduché najít v literatuře jeho precizní odvození. Většinou bývají k dispozici pouze výsledné vztahy. Jde nicméně o algoritmus s výbornými vlastnostmi, který je zřejmě v současné době pro účely potlačení ozvěny využíván nejvíce.

Dobré vlastnosti tohoto algoritmu jsou nicméně vykoupeny značnou výpočetní náročností (uvažujeme vysoké řády filtrů) a dostat se při zpracování pod hranici reálného času vyžaduje kvalitní programování a rychlý počítač. Toto je celkem běžný problém většiny lepších algoritmů a proto bylo v průběhu let vyvinuto značné úsilí na jejich urychlení. V současné době existuje téměř ke každému podobnému algoritmu jeho rychlejší verze. Stejně tak existuje i rychlá verze algoritmu AP označovaná jednoduše FAP (Fast Affine Projection). Získat ovšem o tomto algoritmu kvalitní informace nutné k jeho naprogramování může být téměř nadlidský výkon. Proto se v naší práci musíme spokojit pouze s algoritmem základním.

Pokusme se nyní o jeho alespoň částečné odvození. Vyjdeme opět ze vztahu (3.7), tentokrát ale provedeme jinou (tzv. klouzavou) aproximaci operátoru střední hodnoty a to následující

$$\mathrm{E}[\cdot] = \frac{1}{L}\sum_{k=n-L+1}^{n}[\cdot] \tag{3.22}$$

Vidíme, že náhrada spočívá ve výpočtu aritmetického průměru z celkem L hodnot okolo aktuálního výpočetního kroku n. Funkce (3.7) bude tedy vypadat následovně

$$V(\mathbf{c}_M) = \frac{1}{L}\sum_{k=n-L+1}^{n}\big[e(n)^2\big]. \tag{3.23}$$

Argument n u odchylky $e(n)$ je v pořádku, ačkoliv by zde někdo zřejmě očekával k. Nás ale v kroku n nezajímají odchylky z kroků minulých, my se snažíme nějakým způsobem získat střední odchylku v n-tém kroku. Nejlepším způsobem, jak toho docílit, je „přepočítat odchylky“ z kroků minulých pomocí aktuálního filtru $\mathbf{c}_M(n-1)$, takže (3.23) přejde na

$$V(\mathbf{c}_M) = \frac{1}{L}\sum_{k=n-L+1}^{n}\big[y(k) - \mathbf{x}_M^{\mathrm{T}}(k)\mathbf{c}_M(n-1)\big]^2. \tag{3.24}$$

Derivováním podobně jako u algoritmu LMS opět určíme gradient této funkce, tedy

$$\mathbf{g}_M(n) = -\frac{2}{L}\sum_{k=n-L+1}^{n}\mathbf{x}_M(k)\big[y(k) - \mathbf{x}_M^{\mathrm{T}}(k)\mathbf{c}_M(n-1)\big]. \tag{3.25}$$

V této podobě se ovšem gradient nevyužívá, neboť suma v jeho předpisu je nám pro další úvahy na překážku. Vztah se proto přepisuje do maticové podoby, k čemuž ovšem napřed musíme provést další definice. Tak tedy

$$\mathbf{X}_{M,L}(n) = \big[\mathbf{x}_M(n), \mathbf{x}_M(n-1), \mathbf{x}_M(n-2), \ldots, \mathbf{x}_M(n-L+1)\big] \tag{3.26}$$

je matice vstupních dat rozměru $(M \times L)$ složená ze sloupcových vektorů $\mathbf{x}_M(\cdot)$ a dále

$$\mathbf{y}_L(n) = \big[y(n), y(n-1), y(n-2), \ldots, y(n-L+1)\big]^{\mathrm{T}} \tag{3.27}$$

je sloupcový vektor žádaných výstupních hodnot rozměru $(L \times 1)$. Po ne zcela jednoduchém rozepsání lze ukázat, že vztah (3.25) pro výpočet gradientu lze v maticovém tvaru zapsat jako

$$\mathbf{g}_M(n) = -\frac{2}{L}\mathbf{X}_{M,L}(n)\mathbf{e}_L(n), \tag{3.28}$$

kde $\mathbf{e}_L(n)$ je vektor odchylek daný vztahem

$$\mathbf{e}_L(n) = \mathbf{y}_L(n) - \mathbf{X}_{M,L}^{\mathrm{T}}(n)\mathbf{c}_M(n-1). \tag{3.29}$$

Na výše uvedených vztazích je opět založena celá řada algoritmů, nicméně pro odvození AP algoritmu je třeba jít ještě dále. Algoritmus AP se vyznačuje jedinečnou volbou váhové matice $\mathbf{W}_M$ ze vztahu (3.6), která je dána předpisem

$$\mathbf{W}_M(n) = \left(\frac{1}{L}\mathbf{X}_{M,L}(n)\mathbf{X}_{M,L}^{\mathrm{T}}(n)\right)^{\#}, \tag{3.30}$$

kde operace # značí pseudoinverzi. Důležité je, že tento vztah může být rozepsán na

$$\mathbf{W}_M(n) = L\mathbf{X}_{M,L}(n)(\mathbf{X}_{M,L}^{\mathrm{T}}(n)\mathbf{X}_{M,L}(n))^{-1}(\mathbf{X}_{M,L}^{\mathrm{T}}(n)\mathbf{X}_{M,L}(n))^{-1}\mathbf{X}_{M,L}^{\mathrm{T}}(n). \tag{3.31}$$

Tím jsme téměř u konce. Zavedeme-li ještě označení

$$\mathbf{R}_L(n) = \mathbf{X}_{M,L}^{\mathrm{T}}(n)\mathbf{X}_{M,L}(n), \tag{3.32}$$

můžeme součin váhové matice (3.31) a gradientu (3.28) zapsat ve tvaru

$$\mathbf{W}_M(n)\mathbf{g}_M(n) = -2\mathbf{X}_{M,L}(n)\mathbf{R}_L^{-1}(n)\mathbf{e}_L(n). \tag{3.33}$$

Konečným dosazením tohoto součinu do (3.6) dostáváme výsledek

$$\mathbf{c}_M(n) = \mathbf{c}_M(n-1) + \mu(n)\mathbf{X}_{M,L}(n)\mathbf{R}_L^{-1}(n)\mathbf{e}_L(n). \tag{3.34}$$

Celý algoritmus si později ještě shrneme, aby bylo zřejmé, jak ho vlastně použít. Pojďme si nyní něco říci o výsledcích, které jsme dostali. Pro nás je zajímavá konstanta L, která určuje řád AP algoritmu. Ze vztahů je vidět, že se zvětšujícím se L také nepříjemně poroste výpočetní náročnost. Podívejme se například na vztah pro výpočet matice $\mathbf{R}_L$, který pro běžné hodnoty $M = 1024$ a $L = 10$ představuje součin dvou ohromných matic. Výpočet inverzní matice v posledním vztahu je ve srovnání s tím daleko méně náročný. K tomu je v posledním vztahu ještě další nepříjemný součin a doba výpočtu nám tak narůstá.

Slušného zrychlení lze dosáhnout ve výpočtu matice $\mathbf{R}_L$. Díky speciální struktuře matice $\mathbf{X}_{M,L}$ lze po pracném rozepsání ověřit, že namísto součinu lze výpočet provádět rekurzívně následujícím způsobem

$$\begin{aligned}\mathbf{R}_L(n) &= \mathbf{X}_{M,L}^{\mathrm{T}}(n)\mathbf{X}_{M,L}(n) \\ &= \mathbf{R}_L(n-1) - \mathbf{x}_L(n-M)\mathbf{x}_L^{\mathrm{T}}(n-M) + \mathbf{x}_L(n)\mathbf{x}_L^{\mathrm{T}}(n),\end{aligned} \tag{3.35}$$

přičemž na začátku se provádí inicializace nějakým malým $\delta \approx 10^{-3}$, tj. $\mathbf{R}_L(-1) = \delta\mathbf{I}_L$.

Ještě nám zbývá zmínit se o volbě parametru $\mu(n)$. Pro AP algoritmus se tento parametr pohybuje v rozmezí $0 \leq \mu(n) \leq 1$, přičemž pro $\mu(n) = 0$ adaptace neprobíhá a pro $\mu(n) = 1$ je adaptace optimální z hlediska rychlosti konvergence modelu ke svému vzoru.

Tolik tedy k popisu algoritmu. Na závěr provedeme malou rekapitulaci výpočtu formou tabulky. Jedná se přesně o tytéž vztahy, které jsme zde uvedli, pouze rozepsané do menších částí.

Inicializace
$\mathbf{c}_M(-1) = 0, \quad \mathbf{R}_L(-1) = \delta \mathbf{I}_L$

$$\mathbf{e}_L(n) = \mathbf{y}_L(n) - \mathbf{X}^{\mathrm{T}}_{M,L}(n)\mathbf{c}_M(n-1)$$

$$\mathbf{R}_L(n) = \mathbf{R}_L(n-1) - \mathbf{x}_L(n-M)\mathbf{x}^{\mathrm{T}}_L(n-M) + \mathbf{x}_L(n)\mathbf{x}^{\mathrm{T}}_L(n)$$

$$\mathbf{w}_L(n) = \mathbf{R}^{-1}_L(n)\mathbf{e}_L(n)$$

$$\mathbf{v}_M(n) = \mathbf{X}_{M,L}(n)\mathbf{w}_L(n)$$

$$\mathbf{c}_M(n) = \mathbf{c}_M(n-1) + \mu(n)\mathbf{v}_M(n)$$

Vlastnosti a srovnání algoritmů

Nyní již víme dost, abychom si uvedené algoritmy mohli vyzkoušet a porovnat. Z pochopitelných důvodů se omezíme pouze na algoritmy NLMS a AP, neboť algoritmus LMS ve své nejjednodušší variantě není pro potlačování ozvěny příliš vhodný.

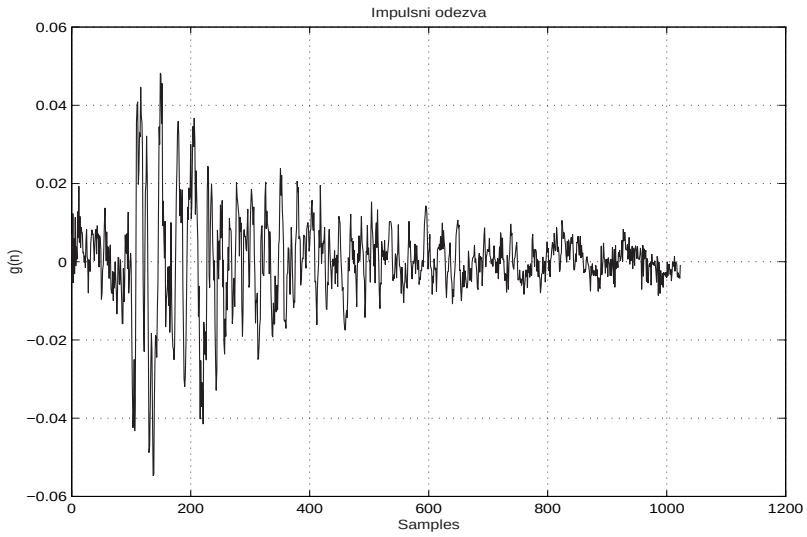
Zajímavou a důležitou vlastností všech algoritmů je jejich rozdílné chování v závislosti na charakteru budícího signálu $x(n)$. Vhodnou charakteristikou budícího signálu pro tento účel může být jeho autokorelační funkce udávající jakousi „soběpodobnost“. Je třeba říci, že všechny algoritmy vykazují nejlepší chování pro málo či vůbec korelované signály, jejichž vhodným příkladem je bílý šum. Lze říci, že je-li budícím signálem právě bílý šum, není mezi adaptivními algoritmy podstatný rozdíl a i velmi jednoduchý algoritmus poskytne stejné výsledky jako algoritmus mnohem složitější.

Čím více je ale budící signál vzdálen bílému šumu, tím horší chování algoritmy vykazují. V uvažovaných aplikacích je budícím signálem řeč a teprve zde se ukáže skutečná kvalita jednotlivých algoritmů. Řeč se totiž skládá z krátkých, téměř periodických, segmentů a prodlev. Sousedící vzorky bývají hodně korelované a ve srovnání s bílým šumem jde proto o velmi nevhodný budící signál.

Popsané vlastnosti si nyní doložíme obrázky. Abychom mohli algoritmy nějakým způsobem hodnotit, musíme umět zjistit, jak dobře a jak rychle se model přibližuje skutečnému systému. Za tímto účelem jsme provedli nepřímé měření impulsní odezvy běžného LEM systému. Obrázek (6) ukazuje tuto odezvu zkrácenou na 1024 hodnot. Dodejme ještě, že pracujeme se vzorkovací frekvencí 8 kHz.

Pohledem na charakteristiku také poznáme další důvod, proč jsme si pro reprezentaci modelu vybrali právě FIR strukturu. Těžko bychom hledali jinou, kterou by bylo možné takovouto odezvu jednoduše vyjádřit. A proč jsme tuto odezvu měřili? Odezva nám posloužila jako vzor skutečného systému, jehož odezvu tím pádem již známe a proto víme, k čemu se má model přibližovat.

Algoritmy jsme se rozhodli otestovat pro dvě různé délky adaptivního filtru (512 a 1024



Obrázek 6: Impulsní odezva měřeného LEM systému

koeficientů). Testování probíhalo následujícím způsobem. Budicí signál jsme nejdříve nechali projít naším modelovým LEM systémem, jednou o délce 1024 koeficientů a poté zkráceným na 512 koeficientů. Tím jsme tedy dostali přesné referenční vstupní a výstupní signály. Poté již zbývalo provést adaptační úlohu a v každém kroku vyhodnotit shodu mezi adaptovaným modelem $\mathbf{c}(n)$ a LEM systémem $\mathbf{g}(n)$. Vhodným měřítkem shody těchto systémů je tzv. *systémová odchylka* (angl. *system error norm*), definovaná vztahem

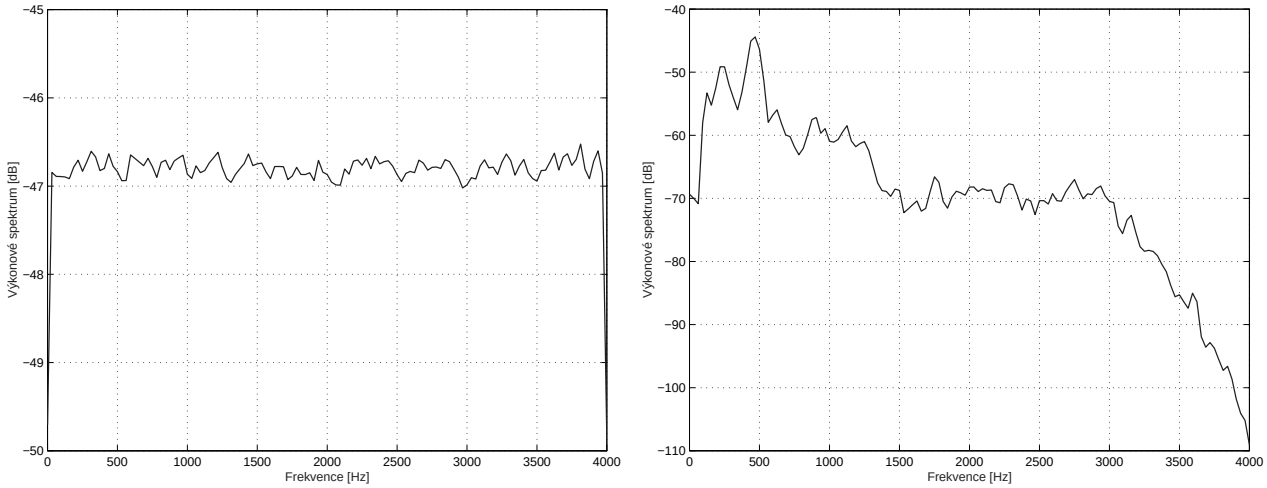
$$10\log(\|\mathbf{c}(n)-\mathbf{g}(n)\|^2) = 10\log\left(\sum_{i=0}^{M-1}\big(c(i)-g(i)\big)^2\right). \tag{3.36}$$

Nepříjemnou vlastností této odchylky je její dosti velký rozsah, neboť v případě ideální shody obou systémů dosahuje hodnoty $-\infty$. Z tohoto důvodu je třeba odchylku trochu upravit. Jelikož v reálných podmínkách je možné dosáhnout systémové odchylky okolo $-50\,\mathrm{dB}$, upravuje se vztah (3.36) tak, abychom v ideálním případě dosáhli právě této hodnoty. Toho dosáhneme jednoduchou úpravou, takže výsledný vztah bude

$$10\log(\|\mathbf{c}(n)-\mathbf{g}(n)\|^2 + 10^{-5}). \tag{3.37}$$

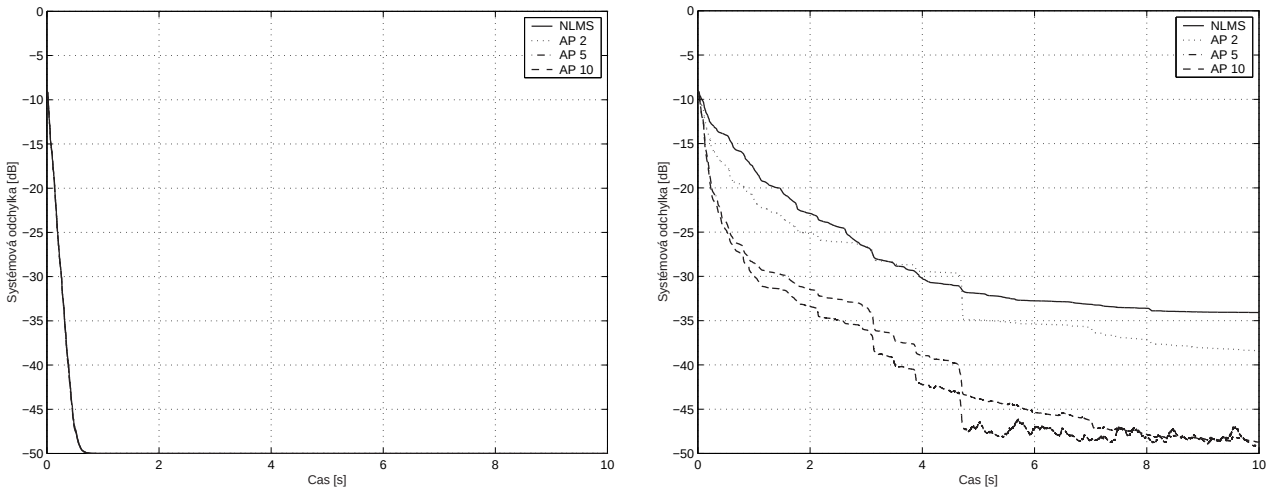
Ještě než si uvedeme časový vývoj systémové odchylky pro jednotlivé algoritmy, řekneme si něco o budících signálech. Již víme, že chování algoritmů je na budicím signálu dosti závislé a proto jsme algoritmy testovali pro dva druhy buzení. V prvním případě byl budícím signálem bílý šum, ve druhém případě běžný řečový signál. Výkonová spektra obou signálů ukazuje obrázek (7).

Nyní se již přesuneme k vlastním testům. Ty jsou shrnuty na obrázcích (8) a (9). Na každém obrázku jsou zobrazeny časové průběhy systémové odchylky pro celkem „čtyři“ algoritmy – NLMS a AP druhého, pátého a desátého řádu. Z obrázků je vidět, že pro bílý šum není mezi algoritmy žádný rozdíl. Teprve pro řečový signál je vidět jasná převaha AP algoritmu ve srovnání s NLMS.



Obrázek 7: Výkonová spektra testovacích signálů – vlevo bílý šum, vpravo řeč

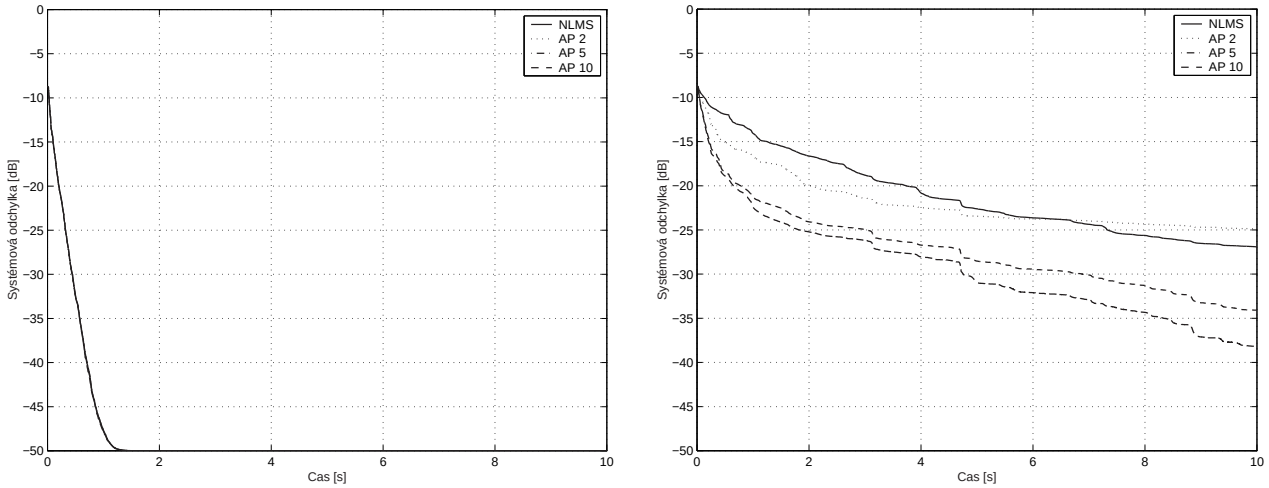
Za povšimnutí rozhodně stojí také rychlost konvergence jednotlivých algoritmů. Vidíme, že tato rychlost celkem logicky s rostoucím řádem filtrů klesá. To je jednoduše řečeno daň za to, že můžeme lépe modelovat danou přenosovou cestu. Rovněž si můžeme udělat představu o vlivu řádu algoritmu AP. Je vidět, že rozdíl mezi pátým a desátým řádem není příliš výrazný a toto platí i pro vyšší řády. Je tedy vhodné nevolit tento řád zbytečně vysoký, neboť vlastnosti algoritmu se příliš nezlepší, naopak výrazně stoupnou výpočetní nároky.



Obrázek 8: Konvergence algoritmů pro bílý šum (vlevo) a řeč (vpravo). Parametry algoritmů jsou $M = 512$ a $\mu(n) = 1$.

Uvedené grafy velmi dobře ilustrují chování obou algoritmů. Zkusíme se na celou věc podívat ještě z jiného pohledu a ukážeme si „záludnost“, která na nás číhá při skutečné realizaci. Od nynějška budeme již pracovat pouze s algoritmem AP a filtrem o délce 1024 koeficientů.

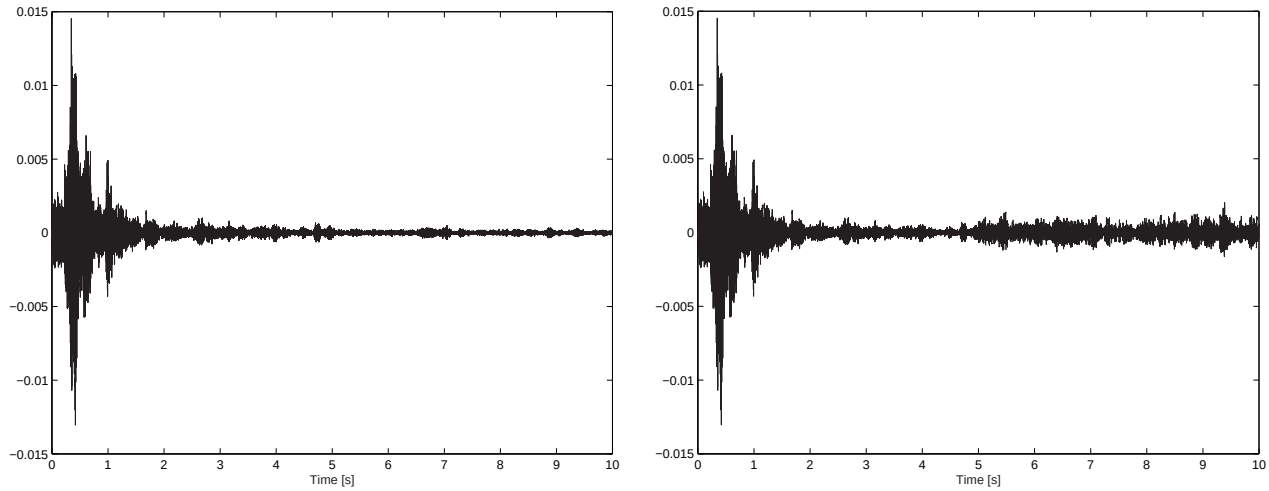
Vraťme se nyní na chvíli k obrázku (4). S kvalitou adaptace velmi úzce souvisí časový průběh rozdílového signálu $e(n)$. Spustíme proto nyní stejný proces jako u předchozího testu



Obrázek 9: Konvergence algoritmů pro bílý šum (vlevo) a řeč (vpravo). Parametry algoritmů jsou $M = 1024$ a $\mu(n) = 1$.

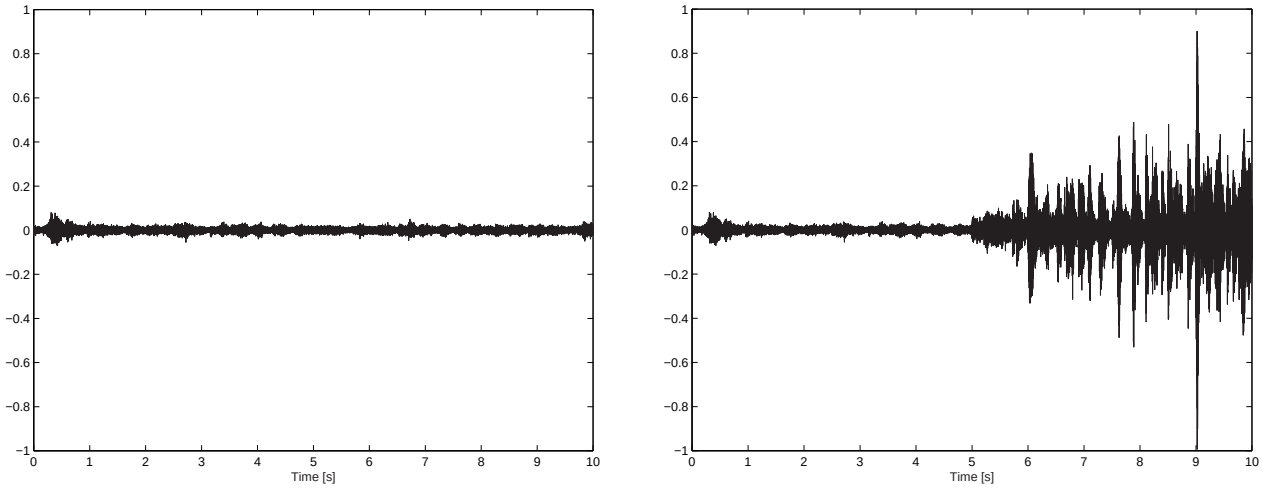
a na tento signál se podíváme. Jeho průběh ukazuje levý obrázek (10). Z obrázku vidíme, že po určité době se model dostatečně přiblíží našemu vzorovému LEM systému a rozdílový signál je téměř nulový (všimněte si měřítka – signál je v rozmezí od -1 do 1). To je očekávaný a velmi dobrý výsledek.

Ukažme si nyní další vlastnost. Spustíme tentýž proces, pouze s tím rozdílem, že po uplynutí pěti vteřin již přestaneme model adaptovat. Průběh signálu $e(n)$ v tomto případě ukazuje pravý obrázek (10). Situace se o něco zhoršila, nicméně pouze mírně, neboť model byl již dostatečně adaptován.



Obrázek 10: Průběhy rozdílového signálu $e(n)$ pro referenční LEM systém

Výše ilustrované, téměř ideální chování celého systému nelze bohužel ve skutečnosti očekávat. Je třeba si totiž uvědomit, že jsme zatím pracovali s modelovými (referenčními) systémy, jejichž parametry se neměnily. Skutečný LEM či podobný systém své parametry ovšem neustále mění, na což musí algoritmus reagovat. Zopakujeme proto předchozí test pro reálný



Obrázek 11: Průběhy rozdílového signálu $e(n)$ pro skutečný LEM systém

systém (použijeme skutečný výstupní signál namísto referenčního). Výsledky ukazují obrázky (11) (všimněte si měřítka). Rozdílový signál na levém obrázku již nelze považovat za nulový, nicméně i toto je slušný výsledek. Situace se podstatně zhorší, přestaneme-li na nějakou dobu model adaptovat. Vidíme, že díky proměnlivosti LEM systému nelze po chvilce náš model za model vůbec považovat, neboť produkuje naprosto odlišný výstupní signál.

Z popsané skutečnosti vyplývá, že je třeba, pokud možno, model adaptovat neustále. To ovšem představuje poměrně veliký problém, neboť v některých případech – např. dvojí řeč, kdy není signál $s(n)$ nulový – je zkrátka nutné adaptaci vypnout. To již ovšem zabíháme do oblasti řízení algoritmů, které bude předmětem následující kapitoly.

3.3 Řízení algoritmů

V této kapitole se budeme zabývat řízením adaptivního algoritmu, které je nezbytné pro zajištění správné funkčnosti celého systému v reálných podmínkách. Opět se zaměříme především na algoritmus AP, popsané principy budou ale obecného charakteru a proto platné či použitelné i pro jiné algoritmy. Je třeba předem říci, že navrhnout dobré řízení je na celém systému pro potlačení ozvěny patrně to nejtěžší a ani v současné době není tento problém úplně vyřešen. Zřejmě nejlepší výsledky lze dosáhnout s využitím neuronových sítí a fuzzy logiky, tím se zde ovšem nebudeme zabývat.

Zkusme nyní vysvětlit, co máme na mysli řízením algoritmu a proč je tohoto řízení třeba. Zatím jsme vždy z hlediska adaptivního algoritmu uvažovali „ideální podmínky“ – signál $s(n)$ byl nulový, tj. komunikující mluvčí vůbec nemluvil, budící signál $x(n)$ naopak nulový nebyl a také jsme vždy předpokládali, že ozvěna skutečně vzniká, k čemuž ale v reálných podmínkách nemusí vůbec dojít. Na tyto situace je třeba pohotově reagovat a chování algoritmu náležitě upravit. Možná se ptáte, jakým způsobem lze algoritmus vůbec řídit. Připomeňme si proto základní formulku pro aktualizaci koeficientů filtru, která má tvar

$$\mathbf{c}_M(n) = \mathbf{c}_M(n-1) + \mu(n)\mathbf{v}_M(n). \tag{3.38}$$

Vidíme tedy, že většina algoritmů je vybavena parametrem μ, tzv. krokem algoritmu,

kterým lze velmi dobře řídit vlastní adaptaci. Položíme-li v libovolném výpočetním kroku $\mu(n) = 0$, zastavíme tím adaptaci. Naopak volbou $\mu(n) \neq 0$ adaptaci sice umožníme, musíme ale uvážlivě zvolit hodnotu tohoto parametru.

Také již víme, že adaptace je prováděna na základě chybového signálu $e(n)$. Je-li tento signál $e(n)$ nulový, adaptace neprobíhá (bez ohledu na parametr μ). Představme si ustálený stav, kdy se shodují model a jeho vzor a signál $e(n)$ je tedy nulový. Narušení tohoto klidového stavu a tedy i nárůst signálu $e(n)$ může mít dvě základní příčiny:

• Promluva mluvčího; signál $s(n)$ bude tedy nenulový a $e(n)$ již nebude vypovídat o shodě mezi systémy. V tomto případě je třeba adaptaci zastavit nebo alespoň zpomalit.

• Změna impulsní odezvy skutečného systému; k této situaci dochází velmi často a v tomto případě je naopak třeba adaptaci uspíšit a model tak co nejrychleji přiblížit jeho vzoru.

△ Je rovněž dobré si uvědomit, že klesne-li úroveň signálu $x(n)$ pod určitou předem stanovenou mez, je rovněž třeba adaptaci zastavit, neboť bez buzení nemůže adaptace vůbec probíhat. V následujících úvahách budeme ovšem buzení $x(n)$ předpokládat.

Jednou z možných cest, jak se s výše uvedenými problémy vypořádat, je navrhnout algoritmus, který zjistí, je-li mluvčí aktivní. Tuto situaci, jak jsme již psali dříve, nazýváme dvojí řeč (double-talk). Z uvedeného je tedy zřejmé, že zachytíme-li tuto dvojí řeč, musíme adaptaci zastavit či zpomalit, což náležitě zajistíme zmenšením parametru $\mu(n)$. V opačném případě zvětšíme parametr $\mu(n)$ na jeho optimální hodnotu (pro AP algoritmus $\mu(n) = 1$), aby adaptace probíhala co nejrychleji. To je samozřejmě pouze základní princip, který lze realizovat mnoha způsoby.

Dvojí řeč není ovšem nijak snadné detekovat, neboť v reálném případě můžeme měřit pouze signál $y(n)$ a tento signál může být nenulový jak vlivem mluvčího, tak i vlivem ozvěny. Už jsme se zmínili, že vhodným prostředkem pro zachycení dvojí řeči může být vzájemná korelace vstupního signálu $x(n)$ a výstupního signálu $y(n)$. Je-li tato korelace „velká“, je signál $y(n)$ z největší části tvořen vzniklou ozvěnou $d(n)$ a je proto velmi pravděpodobné, že mluvčí je potichu a signál $s(n) \approx 0$. Je-li ovšem mluvčí aktivní, bude korelace obou signálů „nízká“. V následující části se budeme korelací zabývat podrobněji, uvedeme některé příklady a postřehy.

Korelace

Z výše uvedeného vyplývá, že v každém výpočetním kroku bude třeba znát korelaci mezi vstupním signálem $x(n)$ a výstupním signálem $y(n)$. Korelace nám zde velmi dobře poslouží jako míra podobnosti mezi těmito signály. Je třeba říci, že existuje celá řada různě definovaných korelací a je nutné si některou vybrat či vyzkoušet více možností. Pro naše účely je nanejvýš vhodné, aby korelace byla určitým způsobem normovaná, tj. aby nabývala pouze hodnot z předem známého intervalu. V publikaci [2] je uvedena vhodná korelační metoda daná následujícím výrazem

$$\rho(n) = \max_{v} \frac{\left| \sum_{i=0}^{W-1} x(n-i-v)\, y(n-i) \right|}{\sum_{i=0}^{W-1} \left| x(n-i-v)\, y(n-i) \right|}. \tag{3.39}$$

Korelace $\rho(n)$ se pohybuje mezi hodnotami 0 a 1, přičemž hodnota 1 značí maximálně korelované signály. V uvedeném vztahu máme k dispozici dva parametry. Tím prvním je časové posunutí mezi signály v. Jelikož právě toto posunutí při zpracování neznáme, je vhodné hledat maximální korelační hodnotu $\rho(n)$ pro různá tato posunutí a zajistit tak robustnost metody. To má ovšem svá úskalí, k čemuž se ještě dostaneme. Druhým parametrem je počet *minulých* vzorků W, ze kterých budeme korelaci počítat. Tento parametr je dosti problematický, neboť je třeba splnit dva protichůdné požadavky. Tím prvním požadavkem je, aby hodnota $\rho(n)$ skutečně vypovídala o podobnosti signálů, k čemuž je potřeba velké množství dat, tj. velké W. Zvolíme-li ovšem W velké, bude hodnota $\rho(n)$ vypovídat více o minulosti než o přítomnosti. To v důsledku povede k tomu, že informace o korelaci obou signálů bude zpožděná a proto např. dvojí řeč nezachytíme dostatečně včas.

Odstranění tohoto problému je poměrně jednoduché, bohužel ale ne vždy použitelné pro jakoukoliv aplikaci. Princip spočívá v tom, že pro výpočet korelace v n-tém kroku nepoužijeme pouze data z minulosti, ale také data z budoucnosti. Toto si samozřejmě můžeme dovolit, nepracujeme-li v reálném čase a máme-li oba signály předem k dispozici. Pracujeme-li ovšem v reálném čase, může pro nás být tento způsob nepřijatelný. Pro jeho realizaci je totiž třeba zavést do celého systému zpoždění, abychom mohli s budoucími hodnotami vůbec pracovat. Záleží potom na aplikaci, bude-li její chod tímto zpožděním nějak výrazně narušen. Přijmeme-li tato omezení, můžeme vztah (3.39) upravit do následující formy

$$\rho(n) = \max_{v} \frac{\left| \sum_{i=0}^{W-1} x(n+Q-i-v)\, y(n+Q-i) \right|}{\sum_{i=0}^{W-1} \left| x(n+Q-i-v)\, y(n+Q-i) \right|}. \tag{3.40}$$

Přibyl nám jeden parametr Q, určující, kolik vzorků z budoucnosti budeme pro výpočet používat. Jeho význam je jistě zřejmý a stojí za povšimnutí, že volbou $Q = 0$ přejdeme ke vztahu předchozímu. Význam ostatních parametrů zůstává samozřejmě stejný.

Poměrně zajímavou otázkou je vlastní výpočet podle uvedených vztahů. Nemůžeme si totiž dovolit ještě více zatížit už tak časově náročný algoritmus AP. Proto je nutné rovněž časově náročný výpočet korelace nějakým způsobem optimalizovat. Nabízíme vám proto následující řešení. Všechna data nutná pro výpočet lze uchovávat v matici o rozměru $W \times V$, kde V je nejvyšší posunutí signálů v, pro které budeme korelaci počítat. Možnou podobu matice ukazuje potom obrázek (12).

Pvky matice představují součin různých vzorků obou signálů. Na pořadí řádků či sloupců této matice nám samozřejmě nezáleží. Dále vidíme, že jednotlivé sloupce vlastně tvoří data ze vztahu (3.40) pro různá posunutí v. To podstatné ovšem je, že v každém výpočetním kroku je třeba určit pouze jeden (první) řádek této matice, což představuje velkou úsporu času. Ze struktury matice vidíme, že vložený řádek v následujících krocích postupně „stárne“ a časem se z matice vytratí. Pro efektivní výpočet je ještě třeba v paměti udržovat (a v každém kroku aktualizovat) vektory (o délce V) představující součty, resp. součty absolutních hodnot všech řádků. Výpočet korelace podle vztahu (3.40) je potom již velmi jednoduchý.

y(n+Q) x(n+Q)	y(n+Q) x(n+Q-1)	y(n+Q) x(n+Q-2)	y(n+Q) x(n+Q-3)	. . .	y(n+Q) x(n+Q-V+1)
y(n+Q-1) x(n+Q-1)	y(n+Q-1) x(n+Q-2)	y(n+Q-1) x(n+Q-3)	y(n+Q-1) x(n+Q-4)	. . .	y(n+Q-1) x(n+Q-V)
y(n+Q-2) x(n+Q-2)	y(n+Q-2) x(n+Q-3)	y(n+Q-2) x(n+Q-4)	y(n+Q-2) x(n+Q-5)	. . .	y(n+Q-2) x(n+Q-V-1)
y(n+Q-3) x(n+Q-3)	y(n+Q-3) x(n+Q-4)	y(n+Q-3) x(n+Q-5)	y(n+Q-3) x(n+Q-6)	. . .	y(n+Q-3) x(n+Q-V-2)
. . .	. . .	. . .	. . .	. . .	. . .
y(n+Q-W+1) x(n+Q-W+1)	y(n+Q-W+1) x(n+Q-W)	y(n+Q-W+1) x(n+Q-W-1)	y(n+Q-W+1) x(n+Q-W-2)	. . .	y(n+Q-W+1) x(n+Q-W-V+2)

Obrázek 12: Matice pro výpočet korelace

Řízení s využitím korelace

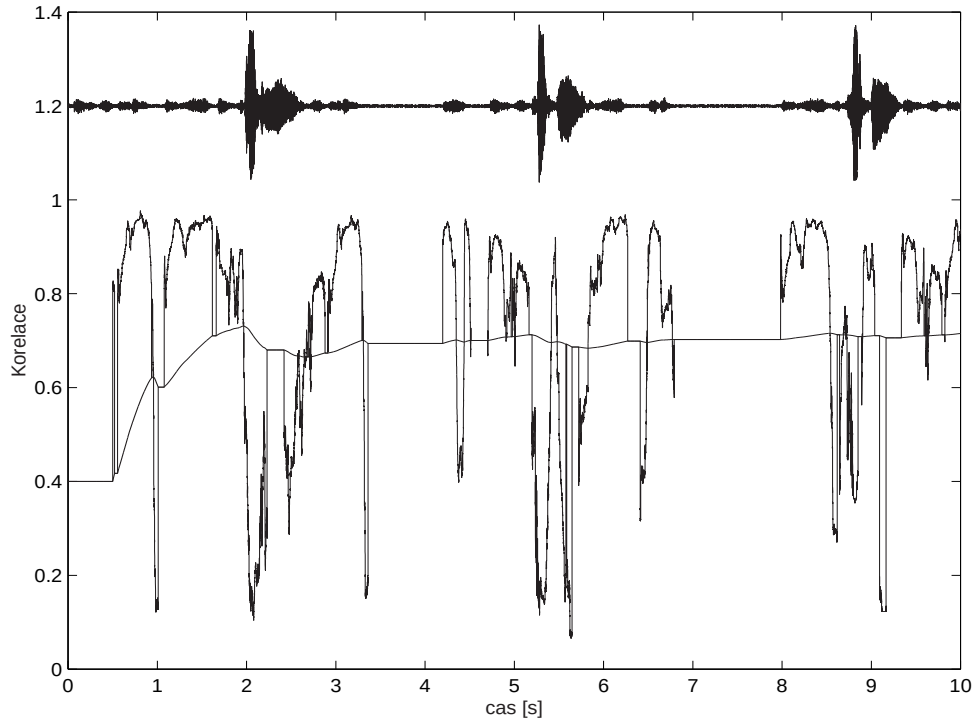
Zde se pokusíme vysvětlit využití popsané korelační metody pro řízení adaptivního algoritmu. Uvedeme zde několik obrázků dokumentujících celou záležitost. Předem čtenáře upozorňujeme, že obrázky obsahují i něco málo navíc, co není v textu přesně popsáno. Jde například o ošetření situace, kdy není dostatečný budící signál $x(n)$ nebo o počáteční hodnoty některých proměnných.

Ukažme si nejdříve některé vlastnosti korelace. Obrázek (13) ukazuje na konkrétním příkladu výstupního signálu $y(n)$ (a samozřejmě i příslušného signálu vstupního) průběh korelace podle vztahu (3.40) (konkrétní parametry jsou $W = 864$, $V = 256$, $Q = 240$). Signál obsahuje slabou (viditelnou) ozvěnu a tři výrazná slova (promluva mluvčího). Na obrázku je také zobrazena průběžná střední hodnota korelace $\bar{\rho}$, o které budeme mluvit později.

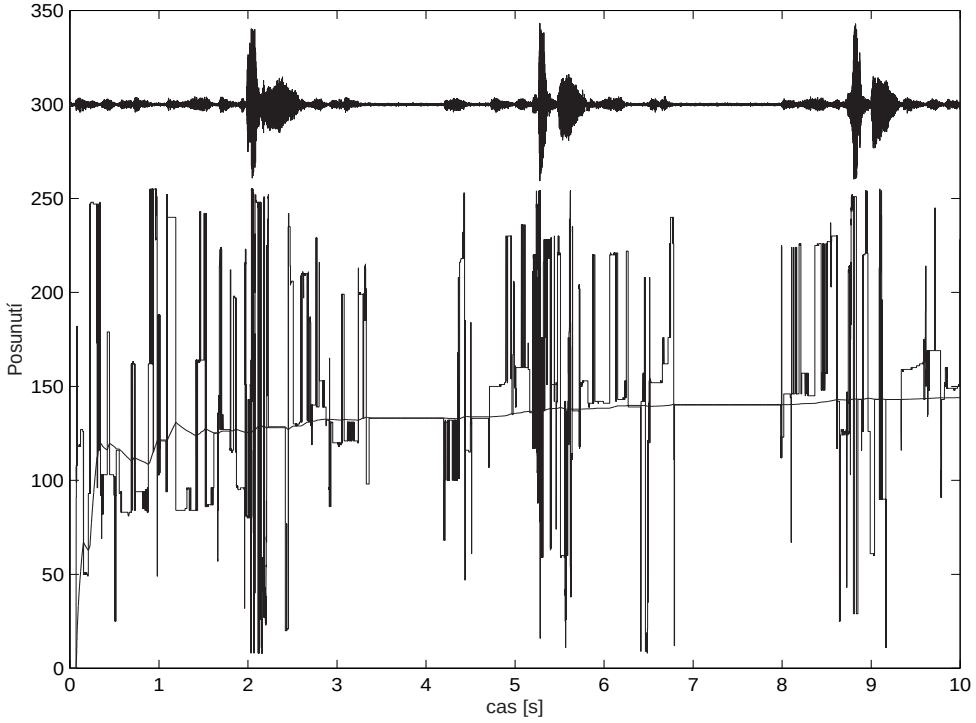
Tato „základní“ korelace není pro řízení ještě příliš vhodná. To je dáno výpočtem podle vztahu (3.40), kde se hledá maximální korelace pro různá posunutí v. Ideální by bylo, aby posunutí v, jemuž odpovídá maximální korelace, bylo v každém kroku stejné. To je naprosto logické, neboť v odpovídá zpoždění mezi signály $x(n)$ a $y(n)$ a toto zpoždění je konstantní. Výpočet ale bohužel ideální není a proto posunutí, pro něž je nalezena maximální hodnota $\rho(n)$, se v každém kroku liší. Jeho průběh ukazuje obrázek (14). Rovněž je vyznačena průběžná střední hodnota $\bar{v}$.

Vidíme, že ideálně konstantní průběh je velmi chaotický a proto ani korelace podle předchozího obrázku nemůže zaručit dobré výsledky. Velmi pozitivní ovšem je, že střední hodnota $\bar{v}(n)$ odpovídá *skutečnému posunutí* obou signálů. Tato střední hodnota je počítána rekurzívně v každém kroku s novým příchozím „vzorkem“ posunutí $v(n)$ podle vztahu

$$\bar{v}(n) = \bar{v}(n-1) + \frac{1}{n}\big[v(n) - \bar{v}(n-1)\big]. \tag{3.41}$$

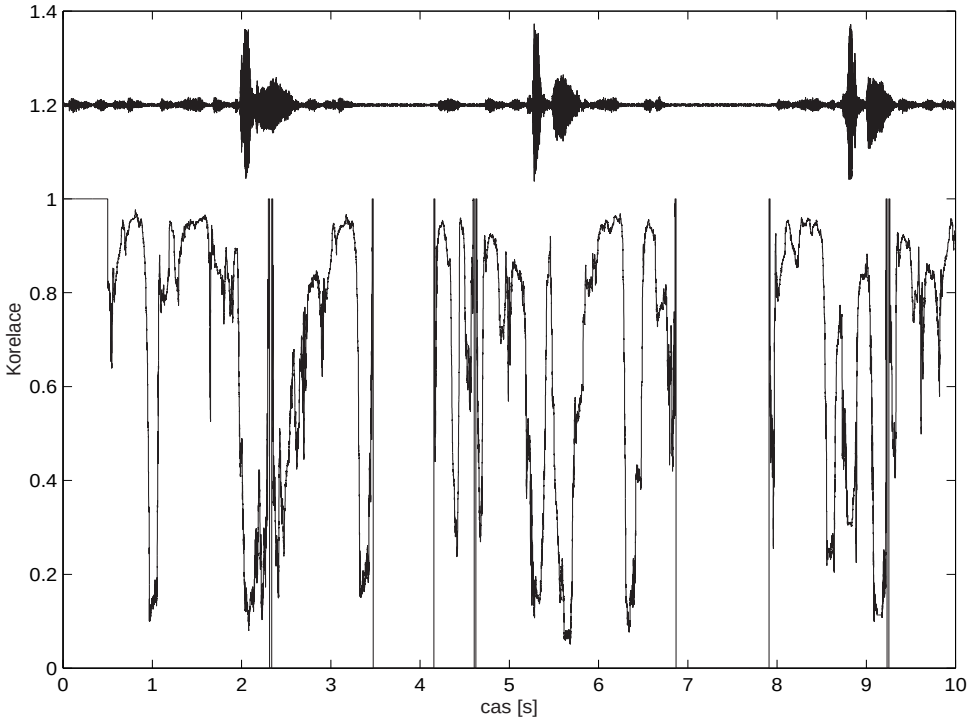


Obrázek 13: Průběh korelace signálů $x(n)$ a $y(n)$



Obrázek 14: Průběh posunutí signálů $x(n)$ a $y(n)$ odpovídajícího maximální korelaci

Nyní můžeme korelační hodnotu podstatným způsobem zkvalitnit. Využijeme faktu, že již přibližně známe skutečné zpoždění mezi signály. Nebudeme proto podle (3.40) hledat maximální hodnotu pro všechna posunutí v, ale pouze pro posunutí okolo hodnoty $\bar{v}$, tedy např. na nějakém volitelném intervalu $\langle \bar{v} - R, \bar{v} + R \rangle$. Výsledek takto upraveného výpočtu (pro $R = 40$) ukazuje obrázek (15).



Obrázek 15: Průběh upravené korelace signálů $x(n)$ a $y(n)$

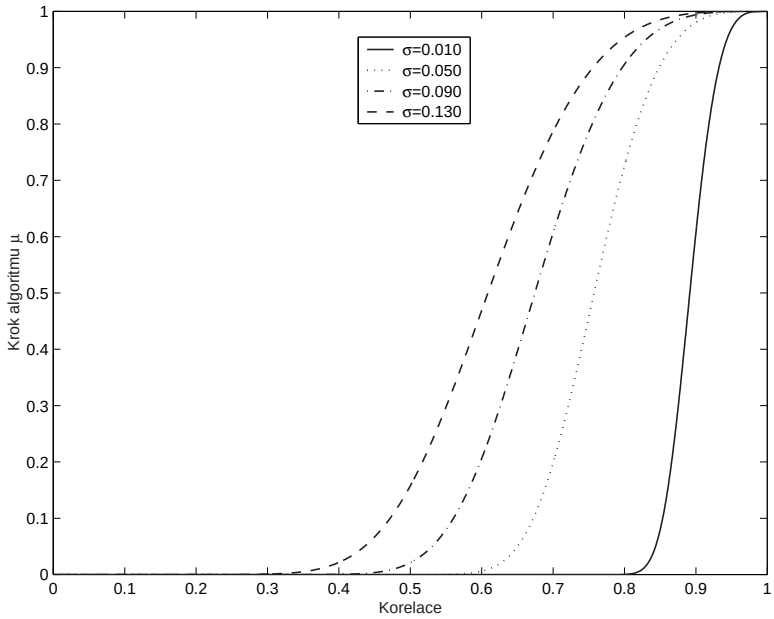
Takováto korelace je již poměrně kvalitní a lze s ní dosáhnout dobrých výsledků. Ideální ale samozřejmě také není. Lze nicméně vymyslet i další podobné variace výpočtu a výsledky tak dále zlepšovat.

Pojďme nyní postoupit o kus dál a zabývat se tím, jak lze uvedenou korelaci využít k řízení. O podstatě řízení jsme již psali dříve a proto víme, že musíme v každém kroku určit parametr μ (pro AP v intervalu od 0 do 1). Korelace ρ nám poskytuje informace o podobnosti signálů a tato podobnost je v přímém vztahu s volbou μ. Je-li korelace nízká, je mluvčí aktivní a adaptaci je třeba zpomalit volbou malého μ. Podobně je tomu v opačném případě. Nejvhodnější tedy bude navrhnout funkci transformující přímo ρ na μ. Volba této funkce je skutečně zásadní, je třeba odhadnout určitou mez mezi malou a velkou korelací, neboli mez situace, kterou považujeme za dvojí řeč. Po dlouhém hledání jsme dospěli k funkci podle vztahu (mírně upravené normální rozdělení)

$$\mu(n) = e^{-\frac{\left[\rho(n) - 1\right]^4}{2\sigma^2}}. \tag{3.42}$$

Více než tento vztah nám o transformační funkci vypoví obrázek (16). Máme zde další volitelný parametr σ, kterým nastavujeme právě „hranici“ mezi aktivním a neaktivním mluv-

čím. Za rozumnou hodnotu lze považovat $\sigma = 0.055$. Obrázek (17) potom ukazuje výsledek transformace korelace z obrázku (15).



Obrázek 16: Transformační funkce mezi korelací a krokem algoritmu

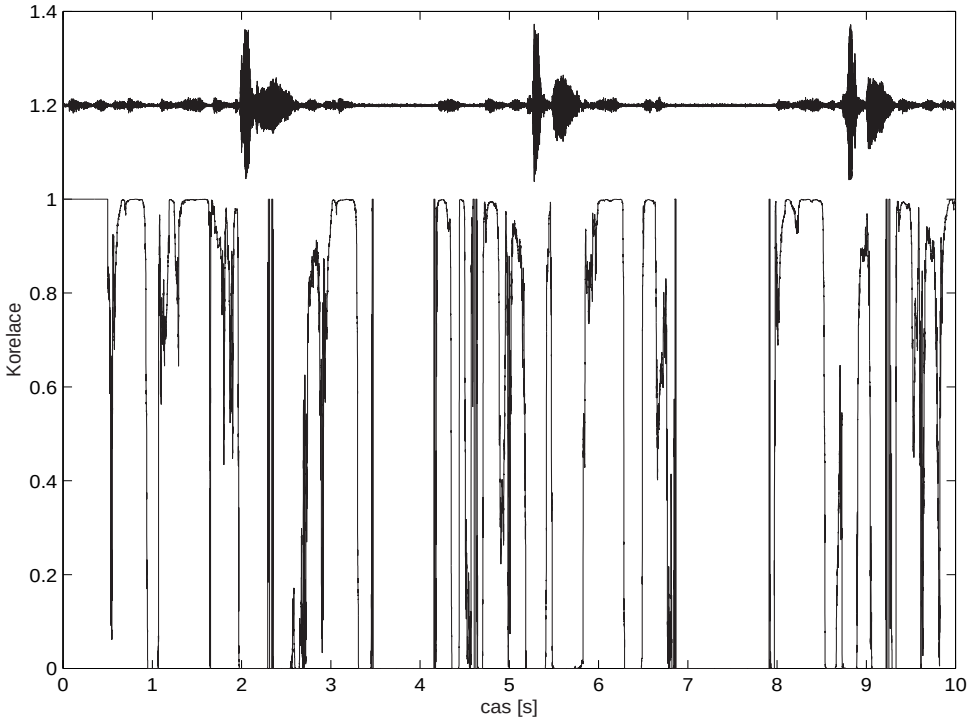
Tím jsme již téměř uzavřeli tuto kapitolu o řízení algoritmů. Je zřejmé, že toto řízení lze realizovat i mnoha jinými způsoby a určitě je stále co vylepšovat. Pokusili jsme se zde pouze popsat a nabídnout metody, které jsme realizovali v naší práci. Na úplný závěr si ještě něco řekneme o poněkud nesnadném „rozběhu“ řídícího algoritmu a detekci ozvěny.

Rozběh a detekce ozvěny

Na samém počátku zpracování se objevuje několik problémů, se kterými je třeba se vypořádat. V první řadě jde o to, že při rozběhu nemáme k dispozici dostatečné množství dat a matice z obrázku (12) není proto zcela zaplněna. Z tohoto důvodu není ani korelační hodnota dostatečně věrohodná a nelze se na ni proto spolehnout. S tím se bohužel musíme po dobu odpovídající zaplnění korelační matice smířit. Podívejme se v této souvislosti na obrázek (13), kde je zpočátku korelace „uměle“ udržována na hodnotě 0.4. U pokročilé korelace z obrázku (15) je situace obdobná.

Dalším nesnadným úkolem při zpracování je zjištění, dochází-li vůbec ke vzniku ozvěny. V celé práci jsme stále předpokládali, že ozvěna skutečně vzniká. Pokud ale k jejímu vzniku nebude docházet, bude adaptivní filtr úplně zbytečný ba dokonce škodlivý. Bude totiž na svém výstupu produkovat nesmyslný signál a výrazným způsobem tak naruší signál přicházející ke zpracování. Tomu je třeba zabránit. My jsme pro tento účel zvolili jednoduchý a účinný postup založený opět na korelaci.

Podívejme se znovu na obrázek (13), kde je vyznačena střední korelační hodnota $\bar{\rho}(n)$. Tato střední hodnota velmi dobře vypovídá o tom, dochází-li ke vzniku ozvěny či nikoliv. Přesáhne-li určitou volitelnou horní mez (cca 0.6), ozvěna docela jistě vzniká. Klesne-li naopak pod mez



Obrázek 17: Výsledný průběh řídícího parametru $\mu(n)$

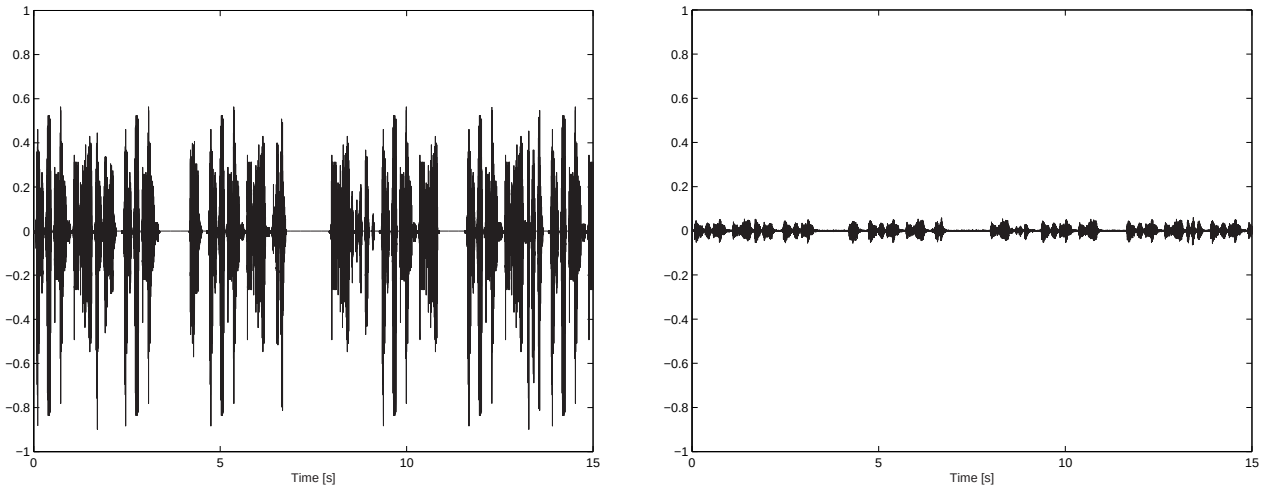
dolní (cca 0.5), ke vzniku ozvěny nedochází. Je tedy rozumné např. během rozběhu čekat, překročí-li $\bar{\rho}(n)$ svou horní mez a dojde-li k tomu, zapojit adaptivní filtr. V opačném případě necháme filtr odpojený a k žádnému narušení signálu tak nedojde.

3.4 Výsledky

V této části se pokusíme shrnout a uzavřít celou tuto kapitolu o ozvěně. Budeme zde, převážně na obrázcích, prezentovat výsledky našeho systému pro potlačení ozvěny realizovaného podle popsaných principů. Pro prezentaci výsledků jsme zvolili dva příklady lišící se pouze velikostí vznikající ozvěny. Aby byly výsledky dobře viditelné, uvedeme zde konkrétní podobu všech signálů vyskytujících se v systému (pro připomenutí odkazujeme na obrázek (4)). Jak víme, skutečně měřitelné jsou ovšem pouze dva – budící signál $x(n)$ a příchozí signál $y(n)$. Čtenáře proto ujišťujeme, že systém pracuje skutečně pouze s těmito signály.

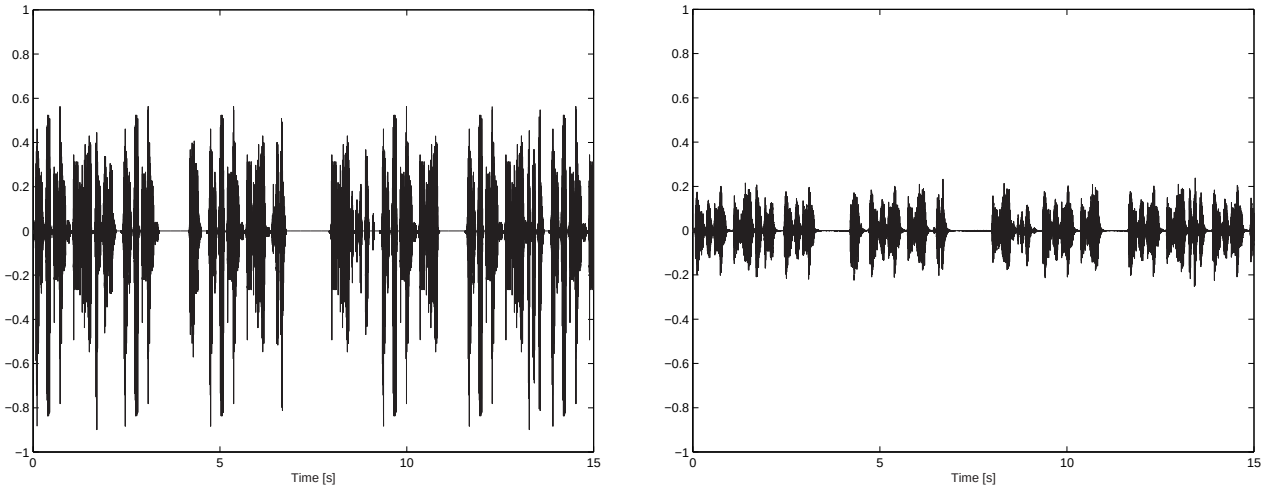
Nejdříve se podívejme na budící signál $x(n)$ a jemu odpovídající signál $d(n)$, který je vlastně vzniklou ozvěnou. Obrázek (18) ukazuje oba signály. Z nich vidíme, že ozvěna je poměrně zřetelná. Nyní se přesuňme ke skupině obrázků (20). Horní obrázek ukazuje promluvu mluvčího – tedy signál $s(n)$ a jeho spektrogram. Pod ním je k vidění měřitelný příchozí signál $y(n)$ (včetně spektrogramu), který je tvořen součtem uvedených signálů, tedy $d(n)$ a $s(n)$. Úkolem systému je „vyčistit“ tento signál od ozvěny a získat tak původní promluvu $s(n)$, která není měřitelná. Výsledek práce systému ukazuje spodní dvojice obrázků. Ještě připomínáme, že systém se na začátku rozbíhá a přibližně jednu vteřinu tak žádné potlačení ozvěny neprobíhá.

Obrázky nám ale zřejmě neřeknou tolik co poslech těchto signálů a případného zájemce si proto dovolíme odkázat na přiložené CD. Pohledem můžeme nicméně usoudit, že vyčištěný signál $\hat{s}(n)$ se poměrně dobře podobá svému vzoru (i spektrogramy jsou si velmi podobné) a systém tak odvedl dobrou práci.

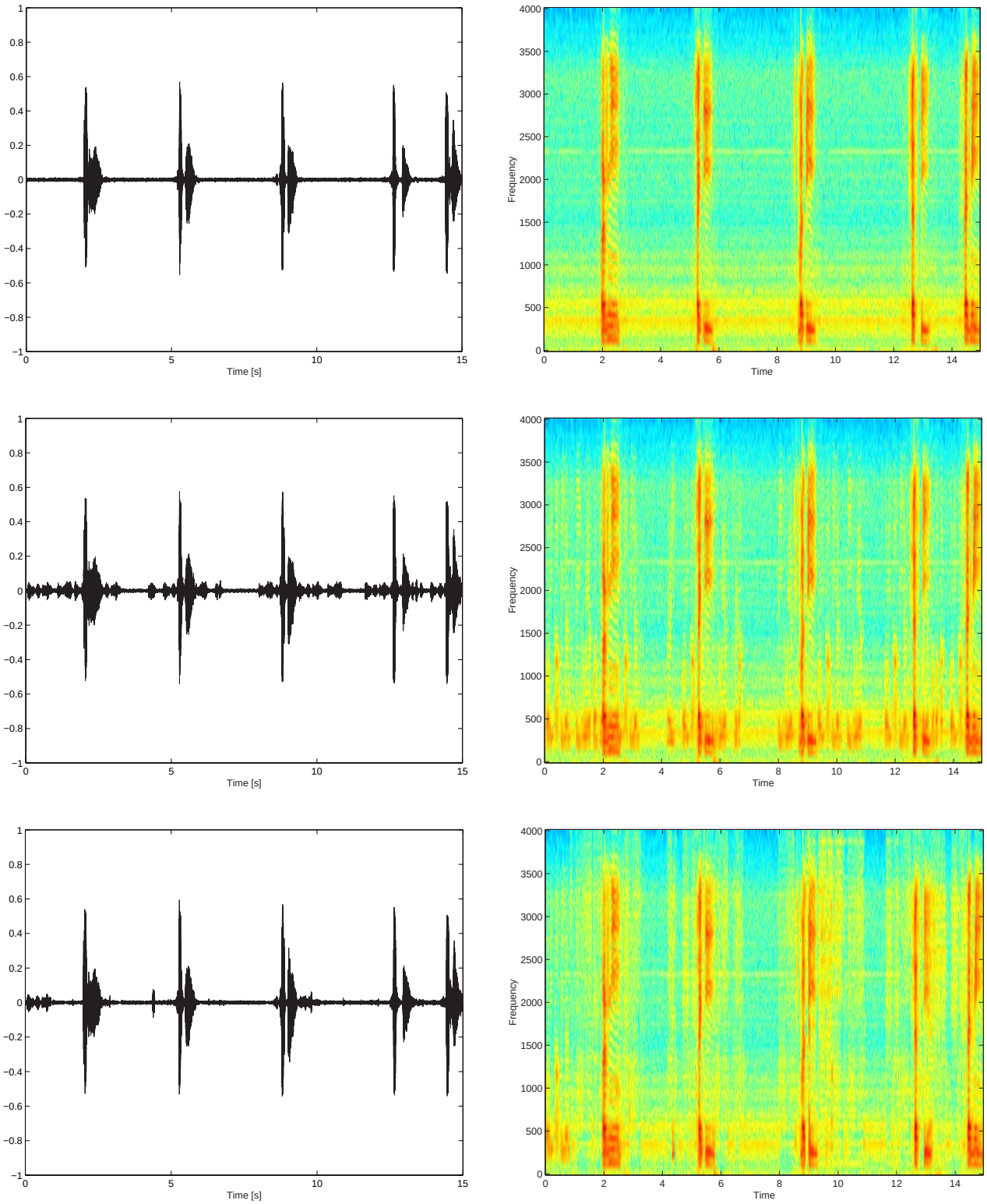


Obrázek 18: Příklad první – budící signál (vlevo) a odpovídající ozvěna

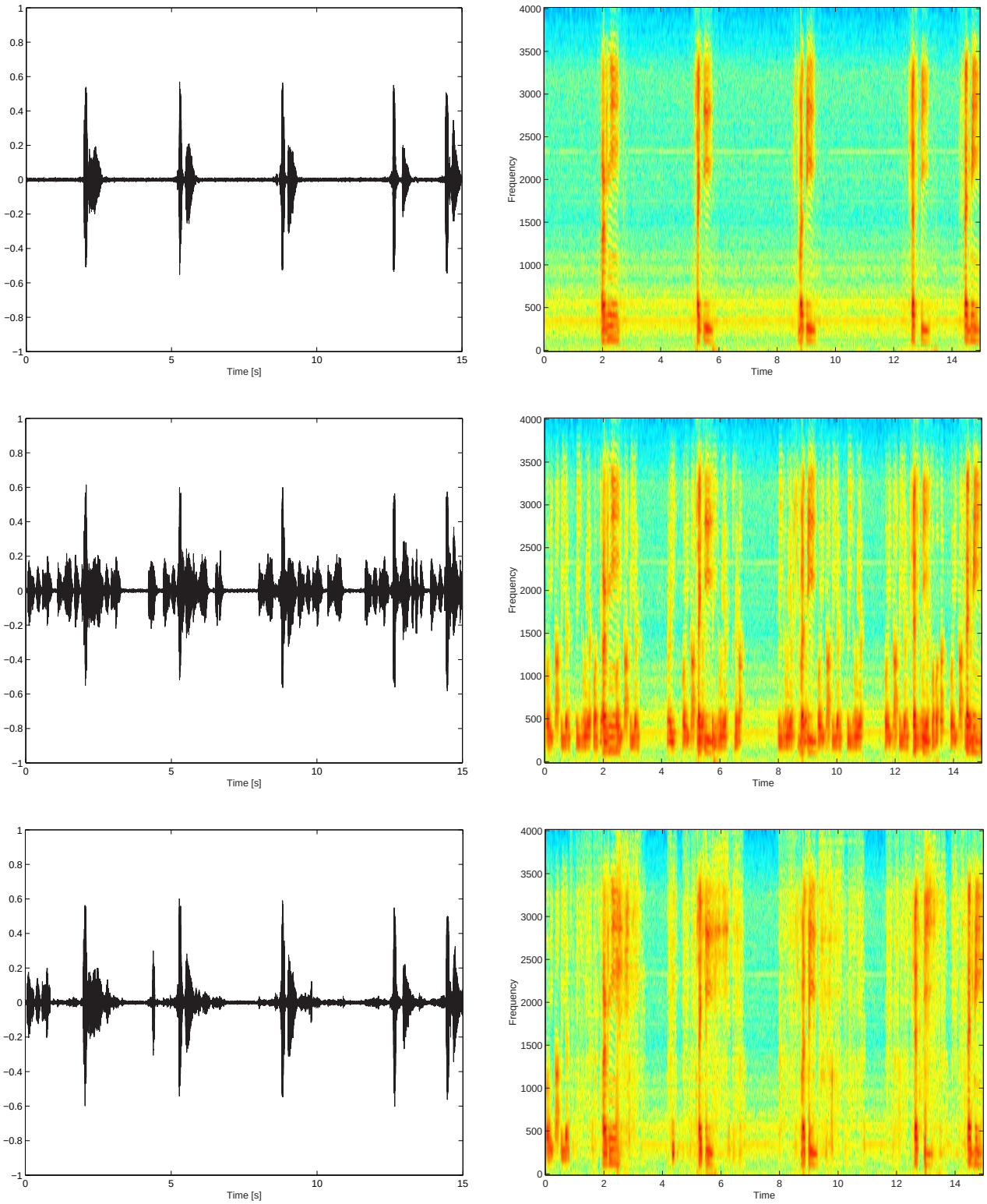
To byl tedy první příklad. Druhý bude o něco méně příznivý. Budící signál a ozvěnu ukazuje obrázek (19). Ozvěna je nyní mnohem výraznější. Ostatní signály, jejichž význam je stejný jako u příkladu prvního, ukazuje skupina obrázků (21). Již méně kvalitní výsledek je způsoben značnou ozvěnou.



Obrázek 19: Příklad druhý – budící signál (vlevo) a odpovídající ozvěna



Obrázek 20: Příklad první – odshora: promluva $s(n)$, příchozí signál $y(n)$ a opravený signál $\hat{s}(n)$; včetně odpovídajících spektrogramů



Obrázek 21: Příklad druhý – odshora: promluva $s(n)$, příchozí signál $y(n)$ a opravený signál $\hat{s}(n)$; včetně odpovídajících spektrogramů

4 Potlačování šumu

4.1 Úvod

Se šumem se setkáváme prakticky ve všech běžných telekomunikačních aplikacích, a to vždy v tom negativním smyslu. Nejlepším příkladem je telefonování, kdy se některý z mluvčích nalézá v prostředí vytvářejícím šum. Zdrojem takového šumu může v uzavřeném prostředí být například topení, klimatizace, hluk počítače atd. V otevřeném prostředí (veřejné telefony) se může jednat o „hluk města“ – silnice, metro, nákupní středisko. A to jsme se ani nezmínili o telefonech mobilních, umožňujících volání téměř odkudkoliv. Všechny tyto příklady představují akustické zdroje šumu, které ale nejsou zdroji jedinými. Šum může vznikat i elektrickou cestou v rámci telefonní sítě.

Z vlastní zkušenosti navíc jistě víte, že komunikace zatížená šumem je pro člověka jistým způsobem unavující a klesne-li tzv. SNR (Signal-to-Noise Ratio – poměr výkonů užitečného signálu a šumu) pod určitou mez, je potom taková komunikace vyloženě nepříjemná.

Přínos metod pro potlačení šumu je tedy evidentní. Těchto metod je dnes celá řada a využívají se v nejrůznějších aplikacích – telefony (pevné i mobilní), hands-free sady (například v automobilech), telekonference, videokonference, rozpoznávání řeči atd. Přínosné je rovněž využití společně s jinými metodami pro zlepšení řečového signálu – máme na mysli hlavně potlačení ozvěny.

Nejrozšířenější metody (nebo spíš variace metody jedné) jsou založeny na *spektrální modifikaci* poškozeného signálu. Jejich obrovskými výhodami jsou především jednoduchost a spolehlivost. Základní myšlenka těchto metod spočívá v převedení signálu z časové do spektrální oblasti, úpravě spektra (potlačení těch frekvencí, kde řeč není přítomna) a zpětné rekonstrukci signálu. Důležitou součástí těchto metod je správná estimace tzv. *obálek* užitečného signálu a šumu. O tom ale později...

Uveďme ještě, že téměř všechny informace jsme čerpali z vynikající (ale těžko dostupné) knihy [1], ve které lze nalézt množství informací týkajících se zpracování akustického signálu. Některé další, více teoretické, informace o zpracování řečových signálu lze také najít v [5].

V následující kapitole si popíšeme jednu z nejznámějších metod potlačení šumu – tzv. Wienerovu filtraci, která slouží jako teoretický základ pro metody založené na spektrální modifikaci.

4.2 Wienerova filtrace

Uvažujme následující situaci. Mějme signál $y(n)$, který je tvořen „hledaným“ nezašuměným signálem $s(n)$ a aditivním šumem $v(n)$, tedy

$$y(n) = s(n) + v(n). \tag{4.1}$$

Naším cílem je samozřejmě odvodit ze známého signálu $y(n)$ neznámý signál $s(n)$. Za tímto účelem je třeba znát nebo co nejpřesněji estimovat výkonová spektra signálů $y(n)$ a $v(n)$. Označme výkonová spektra všech signálů postupně $P_y(\omega)$, $P_s(\omega)$ a $P_v(\omega)$. Dále za určitých teoretických předpokladů[2] platí mezi výkonovými spektry stejná rovnost jako mezi samotnými

[2] jsou-li $s(n)$ a $v(n)$ nekorelované stacionární náhodné procesy

signály, neboli výkonové spektrum zašuměného signálu je jednoduše dáno součtem spektra signálu „čistého“ a spektra šumu, tedy

$$P_y(\omega) = P_s(\omega) + P_v(\omega). \tag{4.2}$$

Z tohoto vztahu je vidět, jak získat (alespoň teoreticky) výkonové spektrum čistého signálu $s(n)$. Prostou úpravou dostaneme

$$P_s(\omega) = P_y(\omega) - P_v(\omega). \tag{4.3}$$

To nám samozřejmě není příliš užitečné, neboť my se snažíme získat signál $s(n)$ a nikoliv jeho výkonové spektrum. Pokračujme tedy dále...

V dalších úvahách budeme rovněž pracovat s Fourierovými transformacemi našich signálů, označme je proto postupně $Y(\omega)$, $S(\omega)$ a $V(\omega)$.

Aniž bychom se pouštěli hluboko do teorie, ukážeme si, jak lze signál $s(n)$ určit nepřímo z jeho Fourierovy transformace. Odhad (estimace) signálu $s(n)$ označme $\hat{s}(n)$ a jeho F. transformaci $\hat{S}(\omega)$. Pan Norbert Wiener ukázal, že takový odhad, který minimalizuje střední kvadratickou odchylku mezi $s(n)$ a $\hat{s}(n)$ je dán vztahem

$$\hat{S}(\omega) = H(\omega)Y(\omega), \tag{4.4}$$

přičemž funkci (filtr) $H(\omega)$ lze odvodit z výkonových spekter podle vztahu

$$H(\omega) = \frac{P_s(\omega)}{P_y(\omega)}, \tag{4.5}$$

což po dosazení vede na vztah

$$H(\omega) = \frac{P_y(\omega) - P_v(\omega)}{P_y(\omega)}. \tag{4.6}$$

Prostým dosazením se již dostáváme k výsledku

$$\hat{S}(\omega) = \left[\frac{P_y(\omega) - P_v(\omega)}{P_y(\omega)}\right] Y(\omega). \tag{4.7}$$

Předchozí vztah nám tedy říká, že spektrum čistého signálu (přesněji odhad spektra) lze získat *aplikací frekvenčně závislého zesílení na spektrum zašuměného signálu.*

Při praktické aplikaci této teorie narazíme samozřejmě na problém určení výkonových spekter signálů. Jelikož tato spektra nebudou nikdy přesně známa, musíme se spokojit s jejich odhadem s využitím Fourierových transformací daných signálů. Vztah (4.7) se potom změní na

$$\hat{S}(\omega) \cong \left[\frac{|Y(\omega)|^2 - |V(\omega)|^2}{|Y(\omega)|^2}\right] Y(\omega). \tag{4.8}$$

Známe-li nyní Fourierovu transformaci čistého signálu, není již velkým problémem (alespoň při číslicovém zpracování) provést transformaci inverzní a získat signál $s(n)$. Předchozí vztah dále předpokládá, že umíme určit F. transformaci šumového signálu $V(\omega)$, což není zdaleka triviální. Je dobré si uvědomit, že pokud v signálu $y(n)$ není řeč přítomna, je spektrum $V(\omega)$ jednoduše totéž co spektrum $Y(\omega)$. Tento poznatek má význam při praktické realizaci, kdy signál $y(n)$ zpracováváme postupně a jsme schopni rozhodnout, je-li řeč v daném časovém okamžiku přítomna či nikoliv.

Parametrická Wienerova filtrace

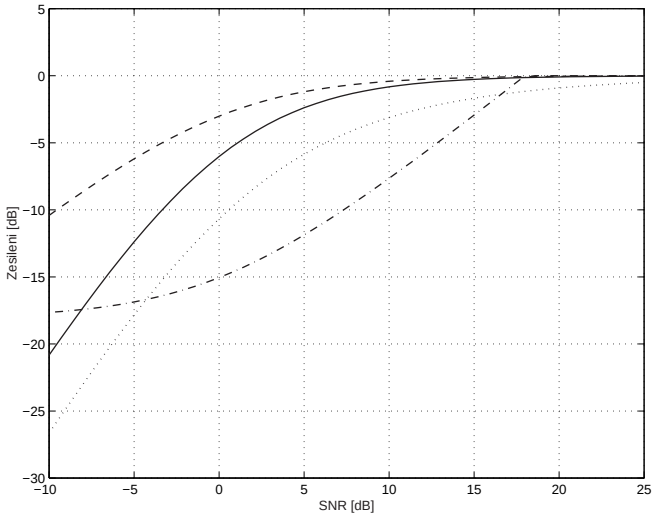
V předchozí části jsme si uvedli jednu z možných podob filtru $H(\omega)$, kterou odvodil pan Wiener a která vychází z požadavku minimalizace střední kvadratické odchylky signálu $s(n)$ a jeho odhadu $\hat{s}(n)$. Tento filtr je ovšem možné navrhovat i podle jiných kritérií a nabízí se tedy otázka jeho zobecnění. Tento zobecněný filtr může vypadat následovně

$$H(\omega) = \left[1 - \left(\frac{|V(\omega)|}{|Y(\omega)|}\right)^{\gamma}\right]^{\beta}, \tag{4.9}$$

kde γ a β jsou volitelné parametry. Volbou ($\gamma = 2$, $\beta = 1$) dostáváme „klasickou“ Wienerovu filtraci.

Zmiňme ještě krátce dvě metody vycházející z jiných principů. Zvolíme-li parametry jako ($\gamma = 2$, $\beta = 1/2$), mluvíme o metodě *výkonového odečítání – power subtraction*, volbou ($\gamma = 1$, $\beta = 1$) lze přejít k metodě *aplitudového odečítání – magnitude subtraction*.

Podrobněji se parametrickou filtrací nebudeme zabývat, alespoň zde ale uvedeme graf, který nám o uvedených metodách poskytne základní informace. Podívejme se nyní na obrázek (22). Ten ukazuje závislost zesílení jednotlivých filtrů na SNR vstupního signálu (v tomto kontextu se mluví o *apriorním*[3] SNR). Tečko-čárkovaná křivka odpovídá filtraci podle řečové aktivity, o které budeme mluvit později. Každá křivka ukazuje, jakým způsobem bude potlačena spektrální složka $Y(k, m)$ v závislosti na SNR, tj. v našem případě na poměru $|S(k, m)|^2/|V(k, m)|^2$. Zajímavé jsou např. hodnoty odpovídající SNR 0 dB, kdy jsou signál a šum ve stejném poměru. Vidíme, že žádná z metod nepotlačí v tomto případě šum o více než 15 dB.



Obrázek 22: Závislost zesílení na SNR; Wiener (plná čára), výkonové odečítání (čárkovaně), amplitudové odečítání (tečkovaně) a řečová aktivita (tečko-čárkovaně)

[3]Zkusme si trochu objasnit pojmy apriorní a aposteriorní. Užíváme-li k blíže nespecifikovanému výpočtu nějakou množinu znalostí, která není úplná (chybí např. znalost nějakého parametru), nazýváme tyto znalosti apriorní. V okamžiku, kdy daný parametr máme k dispozici (doplníme množinu znalostí), mění se naše znalosti na aposteriorní.

Tolik tedy k teoretickým základům, v následující kapitole zmíníme praktickou stránku věcí, o kterých jsme mluvili nyní.

4.3 Spektrální modifikace signálu

Předchozí část nám poskytla poměrně slušné teoretické základy. Jinou otázkou je ovšem realizace těchto principů, neboť ty jsou plně platné pouze pro stacionární signály. Jak víme, řečový signál není stacionární. Tento „nedostatek“ se v praxi vždy obchází tak, že se signál zpracovává po menších částech, v nichž se jeho parametry příliš nemění. Tyto části lze potom za stacionární považovat.

Důležitým nástrojem pro zpracování nám bude diskrétní Fourierova transformace (přesněji její obdoba STFT – short-time Fourier transform), která poslouží při analýze a syntéze řečového signálu. Připomeňme si proto její základy. Libovolný signál $x(n)$ může být reprezentován svou F. transformací danou vztahem

$$X(k,m) = \sum_{n=0}^{N-1} h(n)x(m-n)e^{-j2\pi kn/N}, \tag{4.10}$$

přičemž spektrum je počítáno okolo časového indexu m signálu $x(n)$, k je index diskrétní frekvence, $h(n)$ je analyzační okénko a N je „šířka“ transformace.

Takto získané spektrum navíc můžeme využít pro odhad výkonového spektra signálu, které získáme prostým umocněním absolutních hodnot posloupnosti $X(k,m)$.

Máme-li nyní vhodný nástroj pro výpočet spekter, můžeme trochu pozměnit vztahy z předchozí kapitoly. Tak například vztah (4.4) se s využitím STFT změní na

$$\hat{S}(k,m) = H(k,m)Y(k,m). \tag{4.11}$$

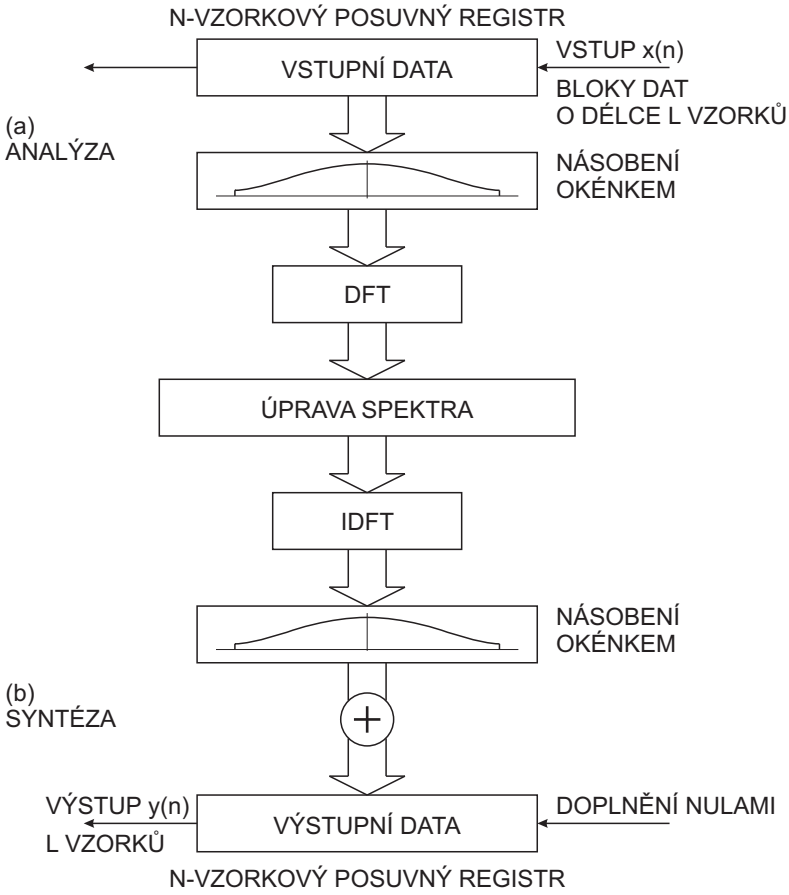
Podobným způsobem se změní i vztahy další a výsledný (4.8) přejde na

$$\hat{S}(k,m) \cong \left[\frac{|Y(k,m)|^2 - |V(k,m)|^2}{|Y(k,m)|^2}\right] Y(k,m). \tag{4.12}$$

Pro nás je ovšem důležité, abychom byli schopni ze spektra signálu provést jeho rekonstrukci. Ta není totiž zdaleka tak triviální, jak by se mohlo na první pohled zdát. Špatná rekonstrukce signálu bude působit nejrůznější rušivé efekty, což samozřejmě není žádoucí. Tím se dostáváme k teorii filtračních bank, která je ovšem příliš hluboká na to, abychom se jí zde podrobně zabývali. Uvedeme si pouze některé základy, které nám pro naše účely plně postačí. Pro podrobnější informace si čtenáře dovolíme odkázat na (opět těžko dostupnou) knihu [7], která se tomuto tématu věnuje dosti podrobně.

Filtrační banka

Filtrační banka je systém, který nějakým způsobem modifikuje spektrum řečového (v našem případě) signálu. Signál je postupně po částech přiváděn na vstup, kde je dále určeno jeho spektrum. Toto spektrum je poté pomocí nějakého filtru $H(k,m)$ změněno (např. potlačen šum) a z takto upraveného spektra je rekonstruován již opravený signál. Příklad jednoduché filtrační banky ukazuje obrázek (23).



Obrázek 23: Schéma filtrační banky

V každém výpočetním kroku je blok L nových vzorků řečového signálu $x(n)$ nasunut do N-vzorkového vstupního registru. Data v tomto registru jsou poté vynásobena okénkem (např. Hammingovým, je ale možné použít i jiné) a transformována pomocí diskrétní F. transformace. V tomto okamžiku se na celou situaci můžeme dívat tak, jakou bychom měli N nových komplexních signálů (pro každou část spektra jeden), ovšem s L-krát nižší vzorkovací frekvencí oproti signálu $x(n)$. Tak například pro vstupní signál se vzorkovací frekvencí 8 kHz, pro $N = 64$ a $L = 16$ tedy dostáváme 64 pásem, každé o šířce 125 Hz a vzork. frekvenci 500 Hz. Dalším krokem ve zpracování je úprava spektra následovaná syntézou (rekonstrukcí). Ta spočívá v provedení inverzní F. transformace, násobení (data se skutečně násobí, ačkoliv by někdo mohl očekávat dělení) pracovního registru okénkem a jeho přičtení k obsahu výstupního registru. Zbývá již pouze vyzvednout L vzorků z výstupního registru a z druhé strany jej doplnit nulami.

Jak vidíme, jde o jednoduchou, výpočetně zcela nenáročnou, a přitom plně dostačující metodu spektrální modifikace, využitelnou v mnoha různých aplikacích.

4.4 Estimace obálek

Zatím jsme v našich úvahách nebyli příliš konkrétní. I když už teoreticky víme, jakým způsobem pozměnit spektrum zašuměného signálu, tj. aplikovat filtr $H(k,m)$, neumíme ještě určit filtr samotný. Samozřejmě, že např. vztah (4.12) nám poskytuje návod, předpokládá ovšem, že umíme určit spektrum šumu $V(k,m)$ (určit spektrum $Y(k,m)$ už umíme). Způsobů existuje mnoho a je dobré si uvědomit, že právě toto je stěžejní problém při realizaci systému pro potlačení šumu.

Jeden z možných způsobů jsme naznačili již dříve. Je-li možné v signálu spolehlivě detekovat okamžiky, kdy řeč není přítomna, je spektrum $V(k,m)$ totéž co $Y(k,m)$. V okamžicích přítomnosti řeči musíme spektrum šumu nějakým způsobem estimovat. Toto lze považovat za základní, ale nejméně vhodný způsob, který se v praxi ukázal býti nepoužitelným.

Podstatného zlepšení v potlačení šumu lze dosáhnout „vyhlazením“ obou spekter $Y(k,m)$ a $V(k,m)$. Jednoduše řečeno jde o jakási průměrná spektra – tzv. *obálky*, které použijeme namísto spekter původních. Označme si tyto obálky pruhem, tj. $\bar{Y}(k,m)$ a $\bar{V}(k,m)$. Existuje opět mnoho způsobů, jak spektra vyhlazovat a i když tím lze mnoho získat, stejně mnoho lze i pokazit. A proč je vlastně třeba spektra vyhlazovat? Pokud bychom to neudělali, byl by náš systém zatížen rušením, které dostalo název *hudební šum – musical noise*. Tomuto rušení bylo v minulosti věnováno mnoho času a názory na jeho příčinu se různí. Intuitivně lze ale přijmout, že nevyhlazené spektrum nezaručuje určitou návaznost při výpočtu filtru $H(k,m)$; filtr se v po sobě jdoucích krocích hodně mění a řeč je doprovázena podivnými efekty. Vyhlazení spektra je proto více než zádoucí. Ukažme si nyní některé způsoby vyhlazování...

Posuvný průměr

Prosté průměrování nás zřejmě napadne nejdříve. Nabízí se tedy způsob průměrování několika minulých hodnot spektra, což můžeme zapsat následujícím způsobem (výpočet obálky $\bar{Y}(k,m)$ je obdobný)

$$\bar{V}(k,m) = \frac{1}{M}\sum_{l=0}^{M-1} |V(k,m-l)|. \tag{4.13}$$

Máme zde jediný volitelný parametr M ovlivňující „hladkost“ výsledného průměru. S tím ovšem také souvisí nutnost uchovávat několik minulých hodnot v paměti počítače. Rovněž jsme se stále nevyhnuli nutnosti nějakým způsobem rozlišit okamžiky přítomnosti a nepřítomnosti řeči. Využití tohoto způsobu je proto dosti omezené.

Jednostranný rekurzivní průměr

Jinou, o něco dokonalejší technikou, je průměrování rekurzivní. Odpadá tedy nutnost uchovávat minulé vzorky v paměti. Výpočet obálky bude vypadat následovně

$$\bar{V}(k,m) = \alpha\bar{V}(k,m-1) + (1-\alpha)|V(k,m)|, \tag{4.14}$$

kde α je koeficient vyhlazení, pohybující se v rozmezí $0 < \alpha < 1$. Z předpisu je vidět, že jde o filtr 1. řádu s přenosovou funkcí

$$F(z) = \frac{1-\alpha}{1-\alpha z^{-1}}. \tag{4.15}$$

Tento způsob vyhlazení přináší více možností než způsob předchozí, i když volba koeficientu α není zrovna snadná. I zde jsme ale stále nuceni nějakým způsobem detekovat řeč, abychom obálku šumu nenarušili vzorkem obsahujícím řeč. U obálky $\bar{Y}(k,m)$ je tento problém samozřejmě obdobný.

Oboustranný rekurzivní průměr

Nyní se konečně dostáváme k podstatně dokonalejší metodě estimace obálek ve srovnání s výše uvedenými. Je sice prakticky stejná jako metoda předchozí, ale drobným vylepšením se podaří získat metodu, která pracuje spolehlivě i bez nutnosti detekce řeči, což je samozřejmě její obrovská výhoda. Předpis jsme již viděli, tj.

$$\bar{V}(k,m) = \alpha\bar{V}(k,m-1) + (1-\alpha)|V(k,m)|, \tag{4.16}$$

jenom koeficient α vybíráme podle toho, jakým směrem se spektrum „vyvíjí“. Přesněji volíme jeden ze dvou koeficientů podle poměru nové příchozí hodnoty $V(k,m)$ a průměru $\bar{V}(k,m-1)$, matematicky zapsáno to vypadá následovně

$$\alpha = \begin{cases} \alpha_a, & \text{jestliže } |V(k,m)| \geq \bar{V}(k,m-1) \\ \alpha_d, & \text{jestliže } |V(k,m)| < \bar{V}(k,m-1) \end{cases}, \tag{4.17}$$

kde indexy u koeficientu α značí angl. slova *attack* a *decay* – volně přeloženo *nárůst* a *pokles*. Přínos celé metody si nyní vysvětlíme. Jelikož máme při výpočtu k dispozici pouze „narušený“ signál $y(n)$ a tudíž pouze spektrum $|Y(k,m)|$, bude předpis pro výpočet obálky šumu vypadat ve skutečnosti následovně

$$\bar{V}(k,m) = \alpha\bar{V}(k,m-1) + (1-\alpha)|Y(k,m)|. \tag{4.18}$$

Vhodnou volbou koeficientů α_a a α_d lze zajistit, že obálka šumu nebude narušena vzorky obsahujícími řeč, o což nám jde především, neboť neumíme spolehlivě rozhodnout, je-li řeč v daném okamžiku přítomna či nikoliv. Abychom docílili správné funkce, volíme $\alpha_a > \alpha_d$. V tomto případě nárůst hodnoty $|Y(k,m)|$ (který odpovídá řeči) nebude mít takový vliv na obálku $\bar{V}(k,m)$ jako pokles této hodnoty a obálka tak bude obsahovat pouze šumové složky.

I druhou obálku $\bar{Y}(k,m)$ lze počítat podobným způsobem, tj. podle předpisu

$$\bar{Y}(k,m) = \beta\bar{Y}(k,m-1) + (1-\beta)|Y(k,m)|. \tag{4.19}$$

V tomto případě je naopak žádoucí, aby $\beta_a < \beta_d$, následkem čehož je obálka více citlivá na úseky s vyšší energií obsahující jak řeč, tak i šum.

Takto získané obálky výrazně zlepší chování kteréhokoliv z uvedených filtrů $H(k,m)$ a jsou dobrým předpokladem pro kvalitní systém. Nezodpovězenou otázkou je ovšem velmi nesnadná volba celkem čtyř koeficientů α_a, α_d, β_a a β_d. K jejich stanovení se ještě dostaneme v kapitole 4.6.

4.5 Filtrování podle řečové aktivity

V této kapitole si ukážeme ještě jednu metodu filtrace, která bude vlastně shrnutím toho, o čemž jsme psali. Stejně jako ostatní metody, má i tato svůj unikátní předpis pro výpočet filtru $H(k,m)$, který realizuje jakýsi „jemný“ detektor řeči. Vztahy jsou následující...

$$\hat{S}(k,m) = H(k,m)Y(k,m), \tag{4.20}$$

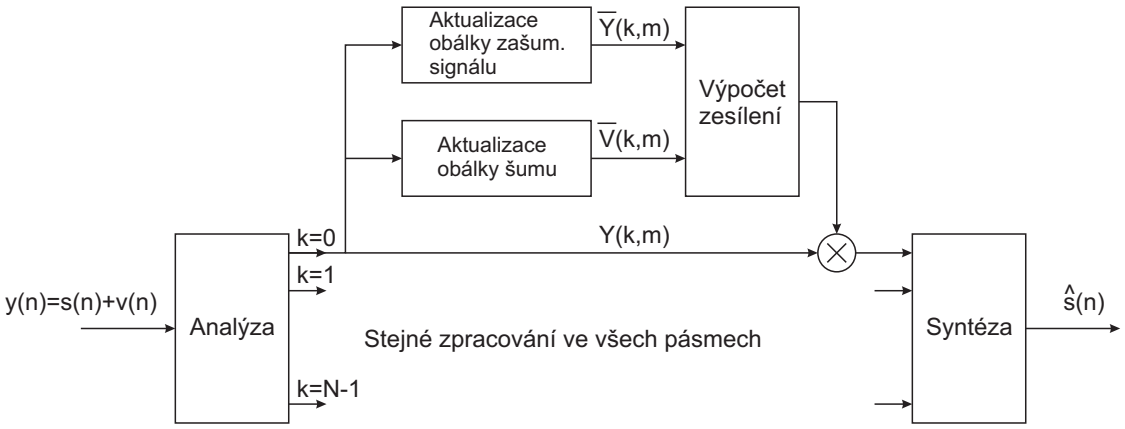
přičemž funkce $H(k,m)$ je definována jako

$$H(k,m) = \min\left[1, \left(\frac{\bar{Y}(k,m)}{\gamma\bar{V}(k,m)}\right)^p\right], \tag{4.21}$$

kde γ je tzv. práh řeči a p expanzní koeficient. Vidíme, že filtr sestává z poměru obálek zašuměného signálu a šumu. I když to není zcela korektní, označuje se tento poměr také jako SNR (obdoba s běžně chápaným SNR je zřejmá), v tomto kontextu navíc jako *aposteriorní*. Ze vztahu je také vidět, že ty složky spektra, jejichž aposteriorní SNR překročí práh řeči, nejsou nijak pozměněny. Naopak ty složky, jejichž SNR nepřekročí práh řeči, jsou potlačeny koeficientem podle uvedeného vztahu. Ještě je dobré vrátit se k obrázku (22), který znázorňuje (tečko-čárkovaně) závislost zesílení tohoto filtru na SNR vstupního signálu.

Za rozumné hodnoty prahu řeči lze považovat interval $5 \leq \gamma \leq 20$, expanzní koeficient p je nějaké přirozené číslo, např. 1. Volba těchto koeficientů je individuální, záleží např. na struktuře filtrační banky nebo na způsobu estimace obálek (konstantách α a β). Určitě se vyplatí trochu s nimi experimentovat.

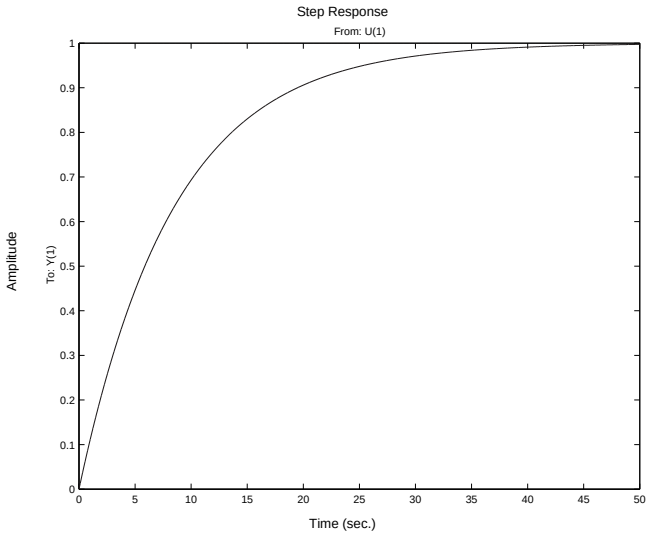
Schéma celého funkčního systému ukazuje obrázek (24). Zašuměný signál v číslicové podobě vstupuje do analyzační části filtrační banky. Zde je vypočítáno krátkodobé frekvenční spektrum $Y(k,m)$, které je použito k aktualizaci obálek narušené řeči $\bar{Y}(k,m)$ a šumu $\bar{V}(k,m)$, metodou oboustranného rekurzivního průměrování. Poté jsou obálky využity k výpočtu zesílení podle vztahu (4.21) (pro každou spektrální složku). Následuje úprava všech složek $Y(k,m)$ těmito zesíleními a pak už zbývá pouze rekonstrukce signálu v syntezační části filtrační banky. Bylo-li vše dobře navrženo, dostáváme na výstupu vyčištěný signál.



Obrázek 24: Schéma systému pro potlačení šumu

4.6 Volba koeficientů pro estimaci obálek

Nyní již konečně víme vše, co je třeba k vytvoření dobrého systému pro potlačení šumu. Velkou roli v tomto systému hrají spektrální obálky, jejichž dobrá estimace je pro celý systém naprosto zásadní. Zde se tedy budeme zabývat volbou koeficientů α_a, α_d, β_a a β_d (samozřejmě uvažujeme oboustranné rekurzivní průměrování). Určit tyto konstanty intuitivně zřejmě nelze, přeci jen jsou v rozsahu od nuly do jedné a význam mohou mít i tisíciny. Můžeme to zkusit jinak. Každá z těchto konstant je určující pro filtr ze vztahu (4.15). Můžeme se tedy místo daného koeficientu pokusit navrhnout časovou konstantu daného filtru a z ní příslušný koeficient vypočítat. Obrázek (25) ukazuje příklad přechodové charakteristiky filtru s časovou konstantou cca 8 s (vzorkovací frekvence vychází z typu filtrační banky, v našem případě $f_s = 8000/16 = 500\,\mathrm{Hz}$).



Obrázek 25: Přechodová charakteristika filtru 1. řádu

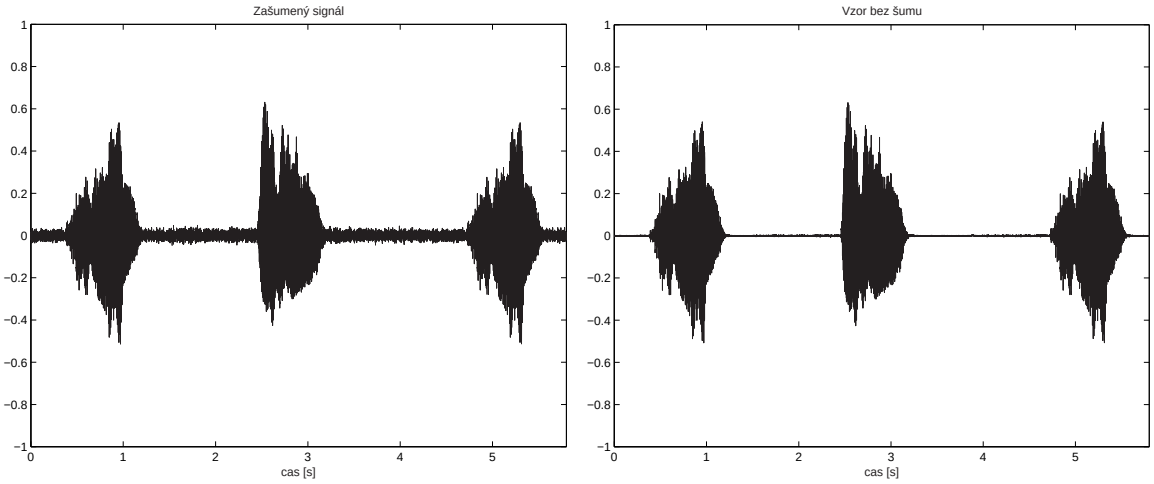
Přepočet mezi koeficientem filtru a časovou konstantou je jednoduchý. Přechodová charakteristika tohoto systému je dána vztahem

$$y(n) = -\alpha\alpha^n + 1, \quad \text{z čehož plyne} \quad \alpha = \exp\left[\frac{\ln(1 - y(n))}{n + 1}\right]. \tag{4.22}$$

Dále víme, že časová konstanta T odpovídá době, kdy přechodová charakteristika dosáhne hodnoty $(1 - e^{-1})$. Po dosazení této hodnoty za $y(n)$ a dále po dosazení $n = T f_s$ lze koef. α již určit. Obrácený přepočet je obdobný.

Tím jsme sice nahlédli trochu „pod pokličku“ této filtrační techniky a snad si i trochu usnadnili hledání vhodných koeficientů, nicméně pro dosažení lepších výsledků lze jít ještě dále. Jako velmi výhodné se jeví přejít k optimalizačním metodám a nechat počítač hledat optimální hodnoty parametrů z hlediska nějakého kritéria. Pro tyto účely je dobré mít k dispozici dvě totožné nahrávky, jednu bez šumu a druhou s šumem, jak ukazuje obrázek (26).

Optimalizace spočívá v hledání takových parametrů α a β, které zaručí co nejlepší shodu mezi „vyčištěným“ signálem a jeho nezašuměným vzorem. Navrhnout hodnotící funkci pro



Obrázek 26: K vysvětlení volby parametrů...

tento účel nemusí být nijak složité. V úsecích obsahujících řeč požadujeme co nejlepší shodu se vzorem, přičemž každou odchylku tvrdě penalizujeme, a v úsecích bez řeči požadujeme nejlépe nulový signál.

Tento způsob vede k velmi dobrým výsledkům, jeho nevýhodou je ale dosti nepříjemná časová náročnost při hledání parametrů.

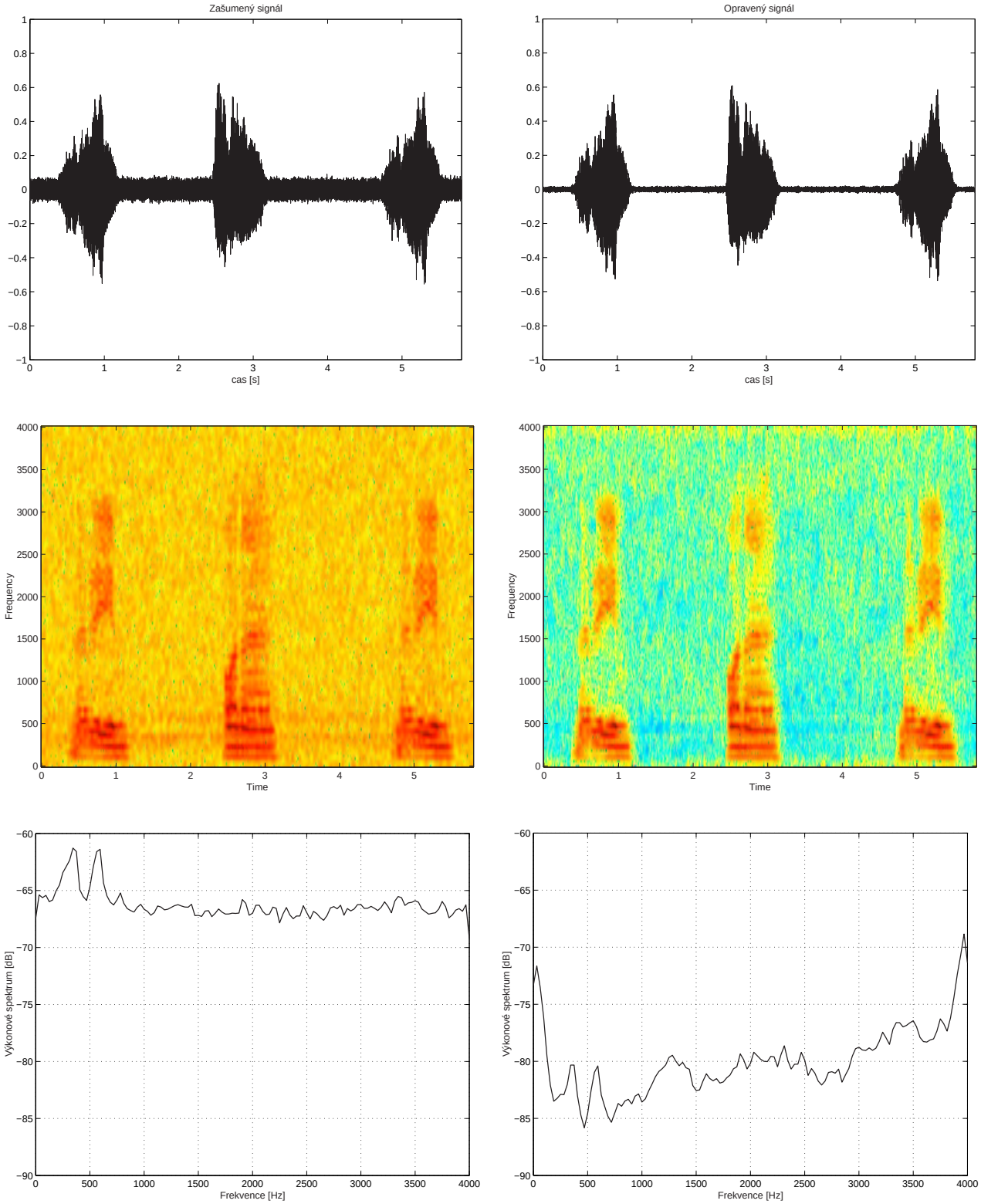
4.7 Příklady a výsledky

Závěrem si ukážeme několik příkladů, které jsou výsledkem celé této kapitoly o potlačení šumu. V obou případech byly koeficienty ze vztahu (4.21) následující: $\gamma = 8$ a $p = 1$.

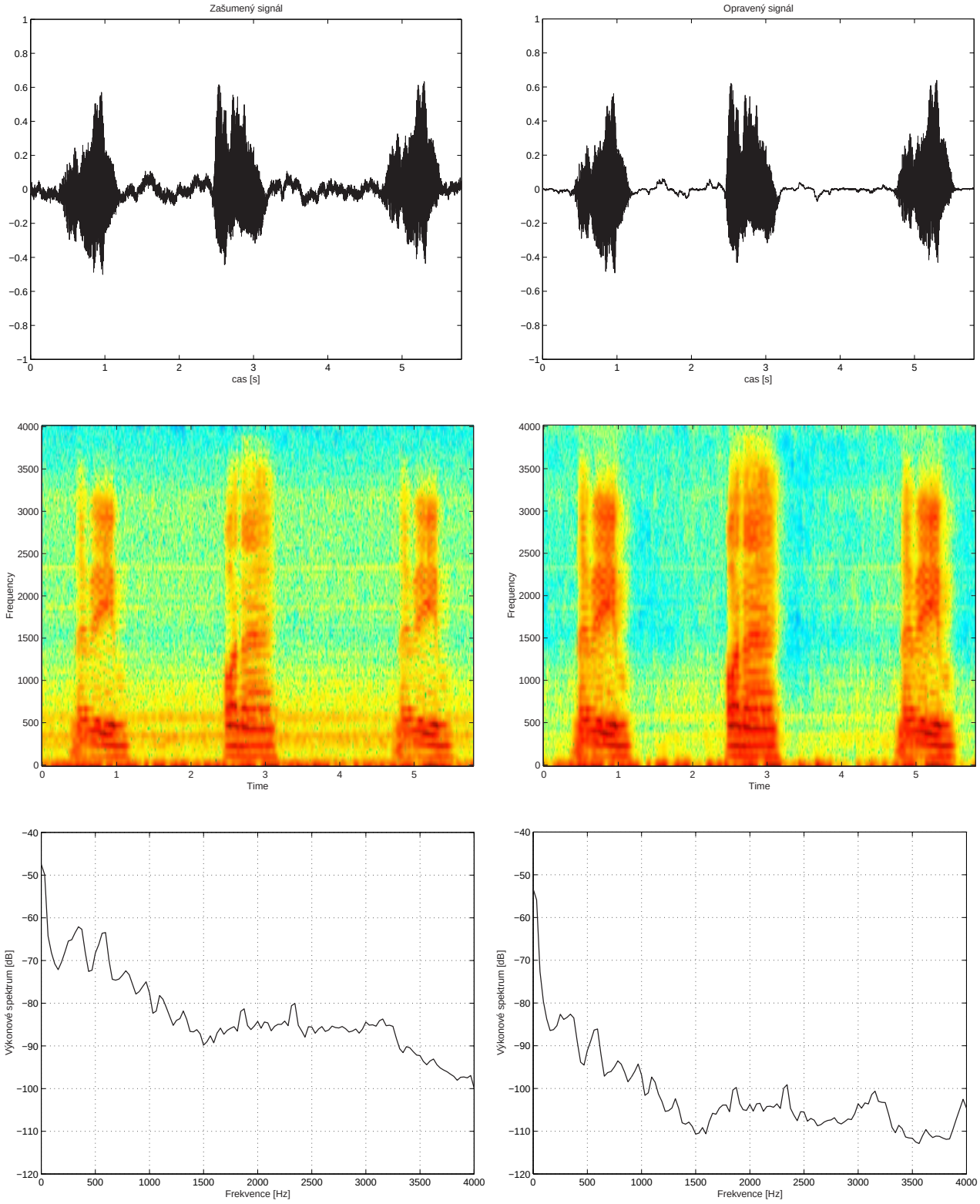
Podívejme se nejprve na obrázky (27). Nahoře je zobrazen výrazně zašuměný signál. Jde o běžný řečový signál se vzorkovací frekvencí 8000 Hz, ke kterému byl přidán náhodný šumový signál. Vedle je zobrazen jeho opravený ekvivalent. Pod nimi jsou dále k vidění jim odpovídající spektrogramy. Poslední dva obrázky ukazují výkonová spektra obou signálů – přesněji pouze částí mezi druhým a třetím slovem, které obsahují pouze šum. Z nich je velmi dobře vidět výrazné potlačení všech spektrálních složek.

Obrázky (28) dokumentují druhý příklad. V tomto případě již nejde o stacionární šum (tzv. Brownův šum) a máme tak možnost vidět logické omezení všech spektrálních metod. Potlačena je pouze *stacionární* složka šumu. Význam ostatních obrázků je stejný jako u předchozího příkladu.

Dodejme ještě něco ke zpracování. Pokud je systém dobře navržen, tzn., že skutečně potlačuje pouze šum a nijak neovlivňuje řeč, je možné (pokud to umožňuje aplikace) nechat „projít“ zašuměný signál systémem několikrát. Výsledky se tak výrazně zlepší.



Obrázek 27: Zašuměný a opravený signál, jejich spektrogramy a výkonová spektra



Obrázek 28: Zašuměný a opravený signál, jejich spektrogramy a výkonová spektra

5 Automatické zpracování a rozpoznávání řeči

5.1 Úvod

V předchozích kapitolách jsme se zabývali převážně úpravou číslicového signálu, který kromě užitečné části obsahoval i část nežádoucí – ozvěnu či šum. Tuto úpravu jsme samozřejmě neprováděli pro nic za nic, nýbrž důvodem našeho konání bylo především poskytnout kvalitní signál pro „vyšší úrovně zpracování“. Pokusme se tedy v této kapitole volně navázat na předchozí a povědět si něco o automatickém zpracování a rozpoznávání řeči – jak jinak než s využitím počítače.

Upozorněme ovšem hned zpočátku na to, že toto téma přeci jen není pro tuto práci stěžejní a není zde tudíž zpracováno stejně podrobně jako témata předchozí. Detailnější informace může čtenář najít např. v [6], odkud jsme mnoho informací čerpali i my.

Zpracování řeči je pojem velmi široký a zahrnuje v sobě celou řadu více či méně řešitelných úloh. Jednou z nejpřitažlivějších je samozřejmě rozpoznávání, kdy se pro nás, pro lidi, tak jednoduchou úlohu snažíme převést na počítač a žádáme po něm, aby „libovolný“ řečový signál převedl do odpovídající textové podoby. Jinou, neméně zajímavou úlohou, je proces opačný, tedy syntéza řeči. Tím jsme ovšem seznam úloh zdaleka nevyčerpali, uveďme ještě např. úlohu automatické identifikace jazyka či rozpoznávání mluvčího. Zajímavým nápadem je také např. využití kamery snímající řečníka, čímž získáme další užitečnou informaci o pohybu rtů. Nechme ale většinu těchto úloh jiným a zaměřme se na „klasické“ rozpoznávání řeči.

Na první pohled nemusí být zřejmé, že i rozpoznávání v sobě zahrnuje více úloh. Tou největší a nejvíce komplikovanou je rozpoznávání souvislé řeči. Je jasné, že v tomto případě na sebe jednotlivá slova plynule navazují, jedno přechází v druhé, aniž je zřejmé, kde je konec prvního a začátek druhého atd. Rovněž každý jazyk má svá výrazná specifika a úlohu je tak třeba řešit pro jiný jazyk téměř od začátku. Zde se proto rozpoznáváním spojité řeči nebudeme zabývat, nicméně některé principy zde popsané jsou využitelné (téměř stěžejní) i pro tuto úlohu.

Další a podstatně jednodušší úlohou je rozpoznávání izolovaných slov. Abychom byli přesní, nemusí jít pouze o slova, ale i o sousloví, krátké věty atp. Podstatné je pouze to, že se v daném signálu vyskytuje právě jedno takové *slovo*. Ještě důležitější ovšem je, že musíme nejprve vytvořit slovník (získat nahrávky od co nejvíce mluvčích) všech slov, která chceme rozpoznávat, což je na celé úloze patrně nejpracnější. Slovník potom představuje pro počítač jistou databázi znalostí, kterou využívá při rozpoznávání. Metod pro řešení této úlohy je samozřejmě více a my si zde dvě z nich popíšeme. Ovšem ještě než to uděláme, musíme si napřed něco říci o vhodném předzpracování signálu nesoucího zatím neznámé slovo.

5.2 Zpracování signálu pro účely rozpoznávání

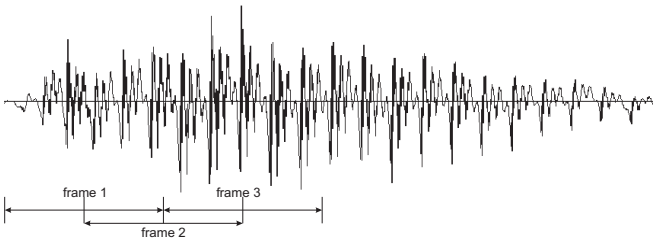
V úvodu jsme již leccos naznačili, nyní postoupíme dále. Řekli jsme, že rozpoznání neznámého slova je založeno na jeho porovnání se všemi referencemi uloženými ve slovníku. Se slovy ovšem nepracujeme v jejich původní číslicové podobě, nýbrž je převádíme do jiné podoby vhodné pro rozpoznávání. Celý proces je naznačen na obrázku (29).

Původní akustický signál je zachycen mikrofonem a převeden do odpovídající číslicové podoby. V současnosti je pro tento účel běžně užívána vzorkovací frekvence 8 kHz a rozlišení

Obrázek 29: Posloupnost akcí při rozpoznávání

16 bitů. Vzhledem k dostupnosti příslušného hardwaru nepředstavuje dnes tento krok žádný problém.

Druhým a velmi podstatným krokem ve zpracování je tzv. segmentace daného číslicového signálu. Ta spočívá v rozdělení signálu na menší části, tzv. framy. Tyto framy se pak již dále nedělí a představují jakousi základní jednotku. Smyslem je rozdělit původní nestacionární signál do menších stacionárních úseků, ve kterých jsou parametry signálu konstantní. Namísto velkého počtu vzorků pak můžeme takovýto úsek vyjádřit podstatně menším počtem parametrů. Vraťme se ještě ke způsobu segmentace. Délka framu se obvykle volí 20 ms, čemuž při vzorkovací frekvenci 8 kHz odpovídá 160 vzorků. Aby navíc byla zachována určitá spojitost po sobě jdoucích framů, je vhodné, aby se framy částečně překrývaly. Obvykle se volí poloviční překryv a nový frame tak začíná každých 10 ms. Z toho lze jednoduše odvodit framovou frekvenci 100 Hz. Ještě dodejme, že segmentaci většinou nemá smysl provádět samostatně, nýbrž dohromady s dalším krokem, kterým je výpočet příznaků nebo též parametrizace.



Obrázek 30: Princip segmentace signálu

Lze říci, že parametrizace je z uvedených akcí klíčová. Neprovedeme-li ji dobře, neposkytne sebelepší klasifikační algoritmus správné výsledky. Princip parametrizace jsme již naznačili – spočívá v *ohodnocení* každého framu určitou sadou *příznaků* (čísel). Frame pak již není vyjádřen vlastními (160ti) vzorky, ale těmito příznaky, kterých bývá podstatně méně (za

rozumný počet lze považovat 10 a více). Redukce vyjádření není ovšem jediným a hlavním přínosem. Důležité je, že příznaky lépe popisují vlastní informaci obsaženou v signálu. Tzn. (alespoň ideálně), že příznaky popisují co bylo řečeno a nikoliv jak to bylo řečeno. Odstraňují tak pro nás nadbytečnou informaci o individualitě mluvčího (výška hlasu, nálada).

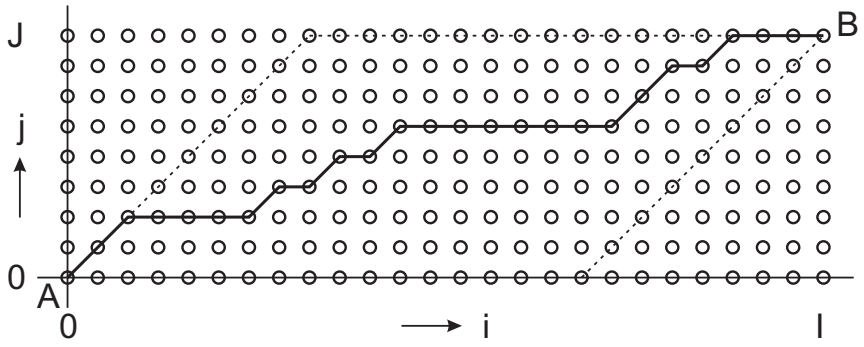
Detaily výpočtu příznaků se zde bohužel zabývat nebudeme a čtenáře proto odkazujeme na publikaci [6]. Tam lze nalézt výpočetní postup pro dnes patrně nejužívanější tzv. *kepstrální* příznaky. Pro začátek lze ovšem vystačit i s příznaky jednoduššími. Jedním z nich je např. energie, která se rovněž využívá při hledání začátku a konce slova. Z ní lze dále odvodit velmi užitečný příznak, tzv. delta-energii, tedy rozdíl energií po sobě jdoucích framů (obdobně delta-delta-energie). Je také možné vyjít při výpočtu příznaků ze spektra framu. Některé spektrální složky mohou posloužit přímo jako příznaky, případně lze celé spektrum vhodně průměrovat. Jiným příznakem může být např. počet průchodů signálu nulou uvnitř jednoho framu.

Shrňme tedy proces parametrizace. Po jeho provedení máme k dispozici pro každý frame určitý počet příznaků neboli příznakový vektor. Používáme-li P příznaků a skládá-li se dané slovo z F framů, můžeme výstup parametrizace uspořádat do matice $F \times P$. Tato matice je novou reprezentací celého slova a dále budeme pracovat pouze s ní. Jsme-li takto „vybaveni“, můžeme v následujících kapitolách přejít k poslednímu článku v řetězci rozpoznávání – ke klasifikaci.

5.3 Bellmanův princip optimality

Nyní si dovolíme malou a pouze zdánlivou odbočku a řekneme si krátce něco o Bellmanovu principu optimality. Na tomto principu totiž stojí dále popisované metody klasifikace a bez jeho pochopení by nemělo další čtení tohoto textu valný smysl. Pro podrobnější informace o Bellmanově principu můžeme čtenáře odkázat na [8], odkud jsme rovněž čerpali.

Vše se pokusíme vysvětlit na konkrétním dvourozměrném případu hledání cesty z jednoho místa A do jiného místa B. Podstatou níže popsaných klasifikačních metod je totiž hledání právě takové cesty. Obrázek (31) znázorňuje celou situaci.



Obrázek 31: K vysvětlení Bellmanova principu optimality...

Na obrázku vidíme síť bodů. Naše cesta prochází právě těmito body. Na vlastnosti cesty můžeme mít celou řadu vlastních požadavků, je nicméně vhodné, aby cesta stále směřovala z výchozího bodu do bodu cílového, tj. nikde se nevracela zpět. Dále si můžeme zvolit, které všechny pohyby z libovolného bodu (i, j) umožníme – na obrázku jsou možné pouze pohyby

$\rightarrow$ a $\nearrow$. Tomu odpovídá i čárkovaně vyznačený prostor, ve kterém se lze nacházet. Není ale problém umožnit i pohyb $\uparrow$. Je asi jasné, že všech možných cest, kterými se lze dostat z bodu A do bodu B, může podle okolností existovat nesmírné množství. Naším úkolem bude nalézt ze všech těchto cest pro nás tu nejlepší.

Abychom tento úkol mohli řešit, musíme umět každou cestu nějakým způsobem ohodnotit. Zaveďme proto funkci $g(i,j)$, která každému bodu sítě přiřadí určitou hodnotu. Tato hodnota může představovat penalizaci za průchod daným bodem. Pokud se ovšem nechceme omezit pouze na body, můžeme obdobným způsobem ohodnotit i přechody mezi jednotlivými body, jednoduše zavedením funkcí $g_{\rightarrow}(i,j)$, $g_{\nearrow}(i,j)$ či $g_{\uparrow}(i,j)$. Každou jednotlivou cestu pak můžeme ohodnotit součtem těchto $g(i,j)$ hodnot podél cesty.

To ovšem není vše. Pro elegantní řešení problému musíme zavést tzv. optimální neboli Bellmanovu funkci $V(i,j)$, která bude v našem případě znamenat ohodnocení *optimální* cesty z libovolného bodu sítě (i,j) do koncového bodu B. Zde jsme narazili na jednu důležitou skutečnost – namísto hledání cesty z bodu A do bodu B, hledáme cestu z libovolného bodu (i,j) do bodu B. Dostaneme tak vlastně řešení pro všechny body sítě a řešení z bodu A si z nich již pouze vybereme. Pro funkci $V(i,j)$ můžeme napsat následující rekurentní vztah (hledáme minimální cestu z hlediska ohodnocení a pohybujeme se ve směrech $\rightarrow$ a $\nearrow$)

$$V(i,j) = \min \begin{bmatrix} g(i,j) + V(i+1,j) \\ g(i,j) + V(i+1,j+1) \end{bmatrix}. \tag{5.1}$$

Tento vztah nám říká, kterým směrem je lepší se z bodu (i,j) vydat – tím, kterému odpovídá minimum. Aby nám byl ovšem skutečně užitečný, musíme si ho vyložit ještě jinak. Říká také, odkud je lepší do bodu (i,j) přijít, vydáme-li se naopak z bodu B do bodu A. Tím jsme vlastně u konce. Máme-li vše zavedeno popsaným způsobem, můžeme při výpočtu postupovat od bodu B po sloupcích k bodu A, v každém bodě sítě určit optimální funkci $V(i,j)$ a rovněž si zapamatovat směr, kterým jsme do daného bodu přišli (neboli směr, kterým se později vydáme). Takto dojdeme až do bodu A a řešení bude úplné.

Zdůrazněme ještě jednou, že celou úlohu lze mnoha způsoby modifikovat a přizpůsobit tak konkrétní situaci. Princip řešení se tím ovšem nijak nezmění. Za zmínku také stojí, že úlohu lze naprosto obdobným způsobem řešit i v opačném směru. Namísto hledání optimální cesty z libovolného bodu (i,j) do bodu B lze hledat cestu z bodu A do libovolného bodu (i,j). Záleží jen na nás, který způsob si vybereme.

5.4 Časové transformace

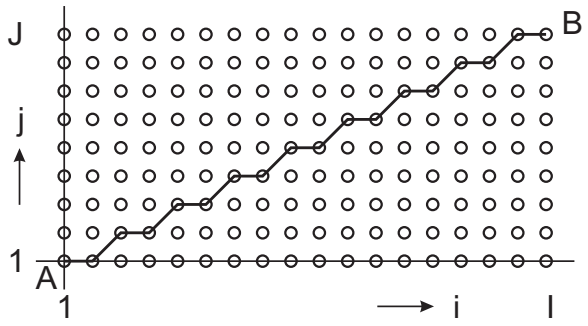
Po malé odbočce se nyní konečně můžeme věnovat vlastnímu rozpoznávání. Zkusme navázat na podkapitolu o zpracování číslicového signálu obsahujícího zatím neznámé slovo. Dostáváme se tedy na další úroveň, ve které jsou všechna slova reprezentována posloupnostmi příznakových vektorů. Zbývá tedy jediné – porovnat neznámé slovo se všemi referencemi a rozhodnout, které z nich se nejvíce podobá.

Za tímto účelem musíme každou referenci jistým způsobem ohodnotit. To první, co nás zřejmě napadne, je porovnat neznámé slovo s referencí frame po framu. Každý frame je dán vektorem příznaků a vhodným ohodnocením může tudíž být součet obyčejných Euklidových vzdáleností mezi framy neznámého slova a reference. Reference s nejmenším ohodnocením pak rozhodne o výsledku rozpoznávání.

Předchozí způsob má samozřejmě háček, neboť jsme tiše předpokládali, že neznámé slovo a reference jsou stejně dlouhé, neboli jsou reprezentovány stejným počtem příznakových vektorů. Tak tomu samozřejmě nebývá a je proto nutné „zarovnat“ slova na stejnou délku. To se děje buď vynecháním některých framů delšího slova nebo zopakováním framů slova kratšího. Oba způsoby lze vhodně kombinovat, jak si nyní ukážeme.

Lineární transformace

Tato metoda srovnávání různě dlouhých slov je velmi jednoduchá a přirozeně proto nelze očekávat příliš dobré výsledky. Princip ukazuje obrázek (32).



Obrázek 32: Ukázka lineární časové transformace – neznámé slovo je orientováno vodorovně, reference svisle

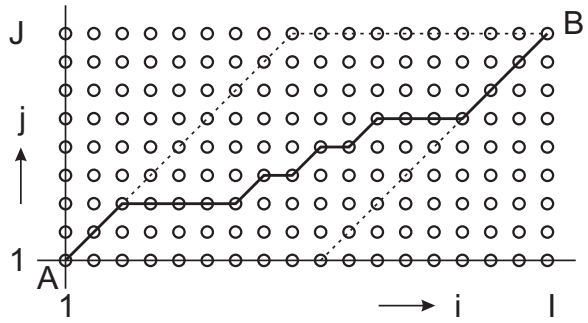
Na obrázku je jakási cesta mezi body A a B, která určuje vzájemné přiřazení mezi framy neznámého slova a reference. Vysvětleme si význam. Neznámé slovo je orientováno podél osy i a skládá se celkem z I framů; v našem případě osmnácti. Reference je obdobně orientována podél osy j; v našem případě má devět framů. Z obrázku vidíme, že jsme prvnímu framu neznámého slova přiřadili první frame reference, druhému framu neznámého slova jsme přiřadili opět první frame reference, třetímu jsme přiřadili druhý atd. Vidíme tedy, že jednotlivé framy reference jsou opakovány a reference je tak „roztažena“ na délku neznámého slova. Ohodnotit tato dvě slova o stejné délce je již velmi prosté. V každém bodě transformační cesty určíme vzdálenost mezi danými příznakovými vektory a výsledné ohodnocení je dáno jejich součtem podél této cesty.

U našeho příkladu je neznámé slovo dvakrát delší než reference. To je pouze náhoda, srovnávat lze slova s libovolnými délkami. Důležité u této *lineární* transformace je pouze to, že onou transformační cestou je právě přímka (v rámci možností vzhledem k rastru).

Nelineární transformace

V následujícím si popíšeme rozšíření předchozí metody, které v důsledku vede k podstatně lepším výsledkům. Budeme postupovat obdobným způsobem, transformační cesta již ovšem nebude nutně přímka. Dlužno podotknout, že existuje celá řada nelineárních metod, lišících se často pouze v požadavcích na vlastnosti transformační cesty. Často se také všechny metody souhrnně označují jako DTW (Dynamic Time Warping).

Příklad nelineární transformace ukazuje obrázek (33). Transformační cesta již není přímka. Navíc v závislosti na našich požadavcích na tuto cestu dochází k omezení „pracovního“ prostoru, který je v našem příkladu vyznačen čárkovaně (volba závisí na nás).



Obrázek 33: Ukázka nelineární časové transformace

Hlavním rozdílem oproti předchozí metodě je způsob určení transformační cesty. Zde se dostáváme k Bellmanově principu. Na jeho základě je totiž možné nalézt optimální transformační cestu, tj. takovou, která má nejmenší ohodnocení. Umožníme tak i pouze lokální zkrácení či prodloužení některých framů, čímž dokážeme co nejlépe zohlednit různě rychlé vyslovení dané hlásky, slabiky či slova. Jednoduše řečeno, obě slova na sebe přiložíme tím nejlepším možným způsobem. Toto opět provedeme pro všechny reference a získáme výsledek.

Ještě nám zbývá pár poznámek k transformační cestě. Na rozdíl od našeho příkladu je vhodné umožnit i přeskočení některého framu reference, což znamená vydat se z daného bodu směrem šikmo, ale s vyšší směrnicí. Dále je například vhodné neprodlužovat libovolný frame reference více než dvakrát. Podobnými omezeními se snažíme zachovat spojitost transformační cesty, což ovšem klade určité požadavky na délky obou slov, které již nemohou být libovolné. Na druhou stranu toto může být i výhoda, neboť nemá velký smysl porovnávat slova výrazně se lišící ve své délce.

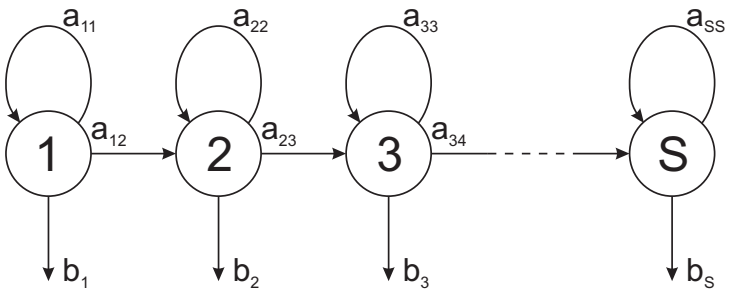
5.5 Skryté markovské modely

V následujícím textu si popíšeme dnes patrně nejpoužívanější metodu klasifikace izolovaných slov. Teorie markovských modelů je poměrně rozsáhlá a může být i velmi složitá. Pokusíme se ale ukázat, že i s několika málo základními znalostmi lze realizovat slušný klasifikátor. Celá metoda se od předchozí ve svém principu podstatně odlišuje, nicméně získané znalosti zde rozhodně využijeme. Připomeňme si napřed krátce předchozí metodu.

Pro každé slovo, které jsme chtěli rozpoznávat, jsme měli „model“, neboli referenci onoho slova od jednoho či více mluvčích. Každé nové neznámé slovo jsme se poté pomocí algoritmu DTW snažili „napasovat“ na všechny reference. Ta nejlépe „padnoucí“ určila výsledek klasifikace. Již z podstaty celé metody vyplývá její malá robustnost. Samozřejmě pracuje spolehlivě např. pro jednoho či malý počet mluvčích. Je ovšem téměř nepoužitelná pro rozpoznávání cizích mluvčích (tzn. těch, kteří nenamluvili slovník). To je dáno jednoduše nesmírnou různorodostí řeči, ze které poté plyne neporovnatelnost neznámého slova s referencemi. Metoda skrytých markovských modelů (dále jen HMM z anglického Hidden Markov Model) řeší právě tento stěžejní problém robustnosti. Jak lze již ovšem tušit, nic není zadarmo a vytvoření

robustního systému si proto žádá nahrávky od značného počtu lidí (nejméně 20, ideálně 40 až 50).

Metoda vychází z následující úvahy – pokud bychom sledovali hodnoty příznaků mezi několika sousedními framy, zjistili bychom, že se příliš nemění. U celé skupiny framů si mohou být příznaky velmi podobné. Nabízí se tedy možnost „sloučit“ tyto sousední framy a reprezentovat je jakýmsi stavem. Nejlépe vše pochopíme podle obrázku (34), kde je zakreslen jeden model určitého slova. Tento model je tvořen několika stavy, přičemž každý stav, jak již bylo naznačeno, zastupuje nějakou skupinu sousedících framů.



Obrázek 34: Struktura modelu

Představme si např. následující situaci. Mějme slovo o deseti framech a čtyřstavový model. První stav může např. zastupovat první tři framy, druhý další dva framy, třetí stav pak např. další čtyři a poslední stav bude pak zastupovat poslední frame. To je jedna z možných situací, ale samozřejmě ne jediná. Těchto zastoupení framů stavy modelu nebo také přiřazení framů stavům modelu existuje celá řada a určit správné zastoupení je klíčovým problémem klasifikátoru. To už ale trochu předbíháme a vraťme se proto ještě k obrázku.

Model se skládá z několika stavů, mezi kterými lze různě přecházet, ovšem nikoliv libovolně. Možné je buď setrvání v aktuálním stavu nebo přechod do následujícího. Vrátit se nelze a nelze ani přeskočit některý stav. Obrázek také napovídá, že ne všechny přechody jsou stejně pravděpodobné. Systém setrvává v s-tém stavu s pravděpodobností a_{ss}, naopak do stavu $s+1$ přejde z s-tého s pravděpodobností a_{ss+1}. Tyto pravděpodobnosti jsou samozřejmě odlišné pro každý stav modelu.

Dále je každý stav popsán ještě dalšími parametry, které si nyní objasníme. Připomeňme si, že každý stav zastupuje několik framů, tzn. zastupuje několik příznakových vektorů. Situaci si nyní trochu zjednodušíme a budeme uvažovat pouze jeden příznak – příznakové vektory se tím změní na skalární hodnoty. Vypočítáme-li z těchto hodnot střední hodnotu μ_s a rozptyl σ_s (pro s-tý stav), získáme další dva velmi významné parametry pro každý stav modelu. Máme-li totiž tyto parametry, můžeme potom pro jakýkoliv frame (popsaný příznakem x) určit míru pravděpodobnosti (vycházíme z normálního rozdělení), že náleží tomu kterému stavu modelu, podle následujícího vztahu

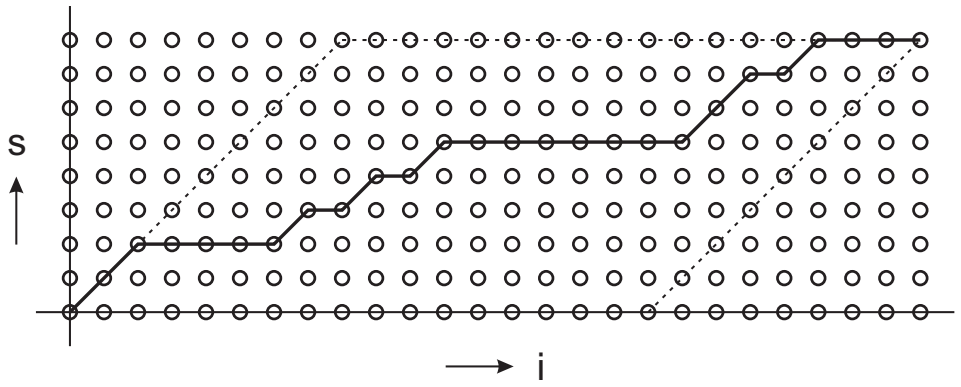
$$b_s(x) = \frac{1}{\sqrt{2\pi}\sigma_s} \exp\Big[-\frac{(x-\mu_s)^2}{2\sigma_s^2}\Big]. \tag{5.2}$$

Tím jsme tedy probrali parametry, které potřebujeme znát pro každý stav modelu. Zatím se nebudeme zabývat otázkou jejich získání – to je věcí trénování, ke kterému se teprve dostaneme. Ačkoliv se to může zdát divné, pro pochopení funkce bude nejlepší předpokládat, že

tyto parametry již máme k dispozici, tedy přeskočit trénování, a přejít k rozpoznávání nebo poněkud přesněji, ke klasifikaci. Situace bude vypadat následovně. Pro každé rozpoznávané slovo musí být k dispozici výše popsaný model se svými parametry. Všechny modely mívají stejný počet stavů (za rozumnou hodnotu lze považovat 12). Klasifikace poté spočívá v „porovnání“ neznámého slova (posloupnosti příznaků) s každým modelem. Model s nejlepším skorem samozřejmě rozhodne o výsledku klasifikace.

Rozpoznávání

Porovnání neznámého slova s některým modelem vychází opět z Bellmanova principu optimality a je proto velmi podobné metodě DTW. V souvislosti s HMM se celý algoritmus označuje jako Viterbiho. Podívejme se nyní na obrázek (35).



Obrázek 35: Přiřazení framů stavům modelu

Z obrázku je vidět, že první frame přísluší prvnímu stavu modelu, druhý frame druhému, třetí až sedmý frame přísluší stavu třetímu atd. Je vidět, že se opět jedná o jakousi „cestu“. Takových cest existuje mnoho a každá z nich udává jiné přiřazení framů stavům modelu. Z každého bodu se můžeme vydat buď vodorovně nebo šikmo (vždy zleva doprava), což odpovídá popsanému modelu – setrvání v daném stavu nebo přechodu do dalšího.

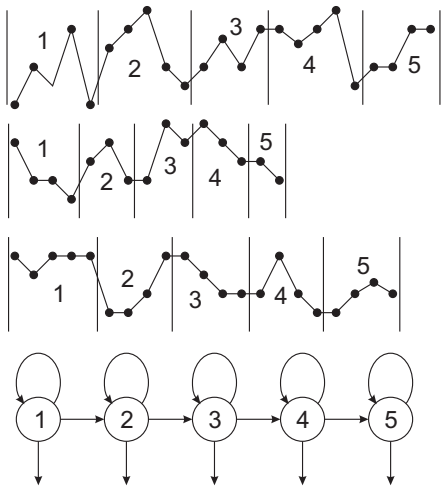
Naším úkolem bude vždy při každém porovnání najít nejlepší možnou cestu. Abychom ovšem ze všech možných cest mohli vybírat nejlepší, musíme umět každou cestu nějak ohodnotit. Ohodnocením bude tentokrát pravděpodobnost, že dané neznámé slovo (popsané příznaky $x_1, x_2, \ldots, x_I$) patří tomu kterému modelu. Tuto pravděpodobnost (vzhledem k určité cestě), vypočteme s využitím výše popsaných parametrů a_{ss}, a_{ss+1} a b_s. Celková pravděpodobnost je dána jako součin všech dílčích pravděpodobností podél cesty. Dílčí pravděpodobnost (pro frame s) je určena součinem míry b_s a pravděpodobnosti a_{ss} nebo a_{ss+1} podle toho, jak jsme se po dané cestě pohybovali. Takže celková pravděpodobnost pro obrázek (35) je

$$p = a_{01}b_1(x_1)a_{12}b_2(x_2)a_{23}b_3(x_3)\cdots a_{33}b_3(x_7)a_{34}b_4(x_8)a_{44}b_4(x_9)\cdots , \qquad a_{01} = 1. \tag{5.3}$$

Umíme-li nyní ohodnotit libovolnou cestu, není už s využitím Bellmanova principu nijak složité najít nejlepší ze všech cest. Pravděpodobnost této cesty je pak vlastně skorem příslušného modelu. Tento postup stačí opakovat pro modely všech slov a nejlepší skore rozhodne.

Trénování

Trénování je proces o něco složitější než rozpoznávání, nicméně klíčovou částí je opět Viterbiho algoritmus – tedy hledání optimálního přiřazení mezi framy a stavy modelu. Co vlastně budeme trénovat? Budeme hledat optimální parametry modelů všech slov. Za tím účelem jsou potřeba nahrávky všech slov od co největšího počtu lidí. Již jsme se o tom zmiňovali – čím více nahrávek, tím větší robustnost systému. Celý proces trénování jednoho modelu si vysvětlíme podle obrázku (36).



Obrázek 36: Začátek trénování modelu

K dispozici máme obecně K realizací některého slova (na obr. jsou tři). Na samém počátku trénování rozdělíme každou realizaci rovnoměrně na tolik částí, kolik stavů má náš model. Na obrázku jsou jednotlivé části očíslovány. Nyní se provede přiřazení framů stavům modelu, tzn. všechny framy z s-tých částí oněch K realizací se přiřadí s-tému stavu. Celkový počet framů připadající na s-tý stav si označíme N_s. Připomeňme ještě jednou, co toto přiřazení znamená – znamená výpočet parametrů (a_{ss}, a_{ss+1}, μ_s, σ_s) pro daný stav. Příslušné vztahy jsou (uvažujeme jediný příznak)

$$\mu_s = \frac{1}{N_s} \sum_{n=1}^{N_s} x_n,$$

$$\sigma_s = \frac{1}{N_s} \sum_{n=1}^{N_s} (x_n - \mu_s)^2,$$

$$a_{ss+1} = \frac{K}{N_s},$$

$$a_{ss} = 1 - a_{ss+1}.$$

To je ovšem pouze začátek trénování. Tím jsme vlastně získali prvotní „odhad“ modelu. Je dobré si všimnout, že pro každé slovo můžeme již nyní určit skore podle vztahu (5.3). Sečtením skore pro všechna slova dostaneme celkové skore modelu, které samozřejmě chceme maximalizovat.

Nyní se opět dostáváme k Viterbiho algoritmu. Jednoduše provedeme tento algoritmus pro všechna slova, čímž dostaneme nové rozdělení framů (dosud bylo rovnoměrné) na stavy modelu. Máme-li všechna slova nově rozdělena, opět vypočítáme parametry modelu a postup opakujeme. Vše provádíme tak dlouho, dokud se celkové skore modelu zlepšuje.

Rozšíření pro více příznaků

V předchozích částech našeho popisu jsme si záměrně zjednodušili situaci a uvažovali pouze jeden příznak popisující každý frame. V reálném případě je ovšem příznaků celá řada. Situaci se ovšem nijak dramaticky nemění. Skalární hodnotu x vyměníme za vektor $\mathbf{x}$ (v p-rozměrném prostoru), následkem čehož se změní některé vztahy. Především je třeba upravit vztah (5.2) pro jednorozměrné normální rozdělení na vícerozměrné

$$b_s(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^p \det \sigma_s}} \exp\Big[-\frac{1}{2}(\mathbf{x}-\mu_s)^{\mathrm{T}}\sigma_s^{-1}(\mathbf{x}-\mu_s)\Big]. \tag{5.4}$$

Určitě stojí za povšimnutí, že μ_s je vektor středních hodnot jednotlivých složek vektoru $\mathbf{x}$ a σ_s je kovarianční matice, jejíž prvky se počítají podobně jako rozptyl podle vztahu

$$\sigma_s^{ij} = \frac{1}{N_s}\sum_{n=1}^{N_s}(x_n^i - \mu_s^i)(x_n^j - \mu_s^j).$$

Kovarianční matice se již sama o sobě blíží matici diagonální, což je dáno statistickou nezávislostí jednotlivých příznaků. Můžeme tedy s klidným svědomím všechny prvky mimo hlavní diagonálu uvažovat jako nulové a vztah (5.4) se tak ještě výrazně zjednoduší.

6 Vyvinutý software

V této kapitole si ve stručnosti popíšeme software, který vznikl v rámci této práce a byl jejím hlavním cílem. Vše je včetně zdrojových kódů v jazyku C k dispozici na přiloženém CD, takže případnému zájemci nic nebrání v ověření funkčnosti či vyzkoušení.

Nosnou část tvoří aplikace pro potlačení ozvěny a šumu. Doplňkovou aplikací je potom plně funkční klasifikátor izolovaných slov založený na markovských modelech. Ve všech třech případech jde o konzolové aplikace ovládané z příkazového řádku, neboť cílem byl především funkční software, nikoliv přítulná okenní aplikace.

Důležité také je, že software pro potlačení rušení je napsán hlavně jako knihovna několika nosných funkcí a měl by tudíž být na úrovni zdrojového kódu použitelný i v některém „vyšším“ systému, což také bylo jedním z cílů.

V následujícím textu si jednotlivé aplikace postupně popíšeme...

6.1 Potlačení rušení

Software pro potlačení rušení se nachází na přiloženém CD v adresáři `ruseni`. Zdrojové kódy jsou dále k dispozici v podadresáři `source`.

Ozvěna

Program pro potlačení ozvěny má název `aec.exe`. Pro spuštění je třeba zadat parametry v následujícím tvaru

```
aec vstup.wav vystup.wav
```

`vstup`, resp. `vystup` jsou nahrávky vstupního budícího signálu $x(n)$, resp. výstupního příchozího signálu s ozvěnou $y(n)$. Obě nahrávky musejí být ve zvukovém formátu `wav` (16 bitů, 8 kHz). Program po dokončení výpočtu uloží svůj výstup do souboru se jménem `vystup@.wav`.

Při používání celého systému je třeba mít na paměti jednu důležitou věc. Přenosová cesta rušivé ozvěny je modelována filtrem o délce 1024 koeficientů (lze změnit). Z tohoto důvodu je **nutné**, aby zpoždění výstupního signálu oproti vstupnímu bylo pokud možno výrazně menší než těchto 1024 vzorků. Jinak by samozřejmě nebylo možné cestu modelovat.

Výpočet programu je dále řízen několika parametry uloženými v souboru `config.txt`, jehož implicitní podoba je uvedena níže.

```
M=1024
L=10
W=864
V=256
Q=240
MEAN_R=40
FULL=864
SIGMA=0.055
SIGMA_V=0.01
BUZENI=0.1
ECHO_HI=0.60
ECHO_LO=0.55
```

Dlužno říci, že parametry lze považovat za optimální a jejich změnu nelze proto příliš doporučit. Význam většiny z nich by měl být po přečtení kapitoly o ozvěně jasný. Při jejich popisu se na tento text budeme odvolávat.

`M`
– počet koeficientů filtru modelujícího přenosovou cestu

`L`
– řád AP algoritmu

`W`
– počet řádků matice pro výpočet korelace (str. 22)

`V`
– počet sloupců matice pro výpočet korelace

`Q`
– posunutí v čase směrem do budoucnosti při výpočtu korelace (str. 21)

`MEAN_R`
– poloměr hledání maximální korelace mezi signály (str. 24)

`FULL`
– definuje dobu (ve výpočetních krocích) rozběhu algoritmu; do této doby algoritmus pouze sbírá data

`SIGMA`
– koeficient převodu mezi korelací a krokem algoritmu (str. 25)

`SIGMA_V`
– koeficient korigující příliš velké změny impulsní odezvy

`BUZENI`
– práh energie určující dostatečný budící signál

`ECHO_HI`
– horní práh střední hodnoty korelace, který zaručuje vznik ozvěny – při jeho dosažení dojde k zapojení filtru (str. 25)

`ECHO_LO`
– dolní práh střední hodnoty korelace, při kterém dojde naopak k odpojení filtru

Šum

Program pro potlačení šumu se jmenuje jednoduše `sum.exe`. Jeho syntaxe je následující

```
sum soubor.wav [xsoubor.txt] [-twice]
```

Soubor `soubor.wav` obsahuje zašuměnou nahrávku. Opět se předpokládá formát `wav` (16 bitů, 8 kHz). Výstupem programu je soubor se jménem `soubor#.wav`. `xsoubor` je volitelný parametr a definuje jméno souboru, který obsahuje postupně hodnoty koeficientů α_a, α_d, β_a a β_d (str. 38). Adresář `ruseni` obsahuje pro tento účel pět předdefinovaných souborů `x1.txt` až `x5.txt`. Implicitně je použit soubor `x5.txt`, jehož parametry lze zřejmě považovat za nejspolehlivější.

Druhým volitelným parametrem je `-twice`, který nařídí dvojnásobné filtrování vstupního souboru.

6.2 Rozpoznávání řeči

Software pro rozpoznávání řeči se nachází v adresáři `rozpoznavani`. Hlavní program se jmenuje `hmm.exe` a lze ho spustit s parametry v následujícím tvaru (při spuštění bez parametrů se na tyto bude sám dotazovat)

```
hmm slovnik mluvci1 mluvci2 mluvci3 ...
```

`slovnik` je textový soubor obsahující seznam všech rozpoznávaných slov. `mluvci`n jsou jména mluvčích, jejichž nahrávky budou rozpoznávány. Všechny tyto nahrávky pro daný slovník se musejí nalézat v podadresáři `rec/mluvci`n. Důležitým souborem je dále `baze.txt` obsahující seznam mluvčích, na kterých proběhne natrénování systému.

Jelikož je použití poměrně intuitivní, vynecháme další popis a ukážeme pouze jednoduchý výstup programu.

```
hmm cisla jan_kos

Nahravam...
===============

rec\mamka_kosova
rec\petr_jerabek
rec\deda_ricar
rec\jirka_novak
rec\pepa_novak
rec\jarda_richter
rec\betka_kunikova
rec\tata_kos
rec\helena_novakova
rec\tomas_mikolanda

Trenuju...
===============

 0.    1642.424  -- 7-->   2877.396
-----------------------------------
 1.    1488.161  -- 8-->   2652.188
-----------------------------------
 2.    1663.516  -- 8-->   2787.094
-----------------------------------
 3.    1548.500  -- 8-->   2734.490
-----------------------------------
 4.    1915.343  --10-->   3646.635
-----------------------------------
 5.    1817.071  -- 8-->   3163.129
-----------------------------------
 6.    2115.845  -- 9-->   3385.266
-----------------------------------
```

```
 7.    1685.038  -- 9-->   2894.078
-----------------------------------
 8.    1698.443  --11-->   3386.729
-----------------------------------
 9.    2605.349  --12-->   4006.097
-----------------------------------

Rozpoznavam...
===============

rec\jan_kos
Dobre   nula       -> nula
Dobre   jedna      -> jedna
Dobre   dva        -> dva
Dobre   tri        -> tri
Spatne  ctyri      -> devet
Spatne  pet        -> devet
Dobre   šest       -> šest
Dobre   sedm       -> sedm
Dobre   osm        -> osm
Dobre   devet      -> devet

 80.00%
```

7 Výsledky diplomové práce

I když je celá řada výsledků této práce obsažena ve vlastním textu, bude vhodné vše nějakým způsobem shrnout a doplnit.

Za nejdůležitější lze považovat software, který v rámci této práce vznikl a který byl popsán v předchozí kapitole. Jde tedy především o systém pro **potlačení ozvěny**. Jeho výsledky lze samostatně vidět na stranách 28 a 29. Dále jde o systém pro **potlačení šumu**, jehož práci dokumentují stránky 40 a 41. Poslední aplikací je plně funkční rozpoznávač izolovaných slov založený na markovských modelech. Veškerý vytvořený software je součástí doprovodného CD. Na další výsledky se nyní podíváme z jiného pohledu.

Nebylo samozřejmě hlavním cílem, aby software pro potlačení rušení pracoval samostatně, ale aby pracoval jako důležitá součást „vyššího“ systému. Tím je především „inteligentní“ dialogový systém umožňující tzv. *barge-in*, neboli nabourávání uživatele do otázky počítače. Další výsledky této práce lze proto ukázat prostřednictvím testu, který má ověřit, je-li potlačení nežádoucího rušení dostatečné a vede-li k lepším výsledkům **rozpoznávače** mluvené řeči, který je nosnou částí dialogového systému.

Pro účely testu byly vytvořeny nahrávky izolovaných slov narušené ozvěnou a šumem. V první řadě šlo o 100 nahrávek se skutečnou akustickou ozvěnou. Tyto nahrávky byly poté předloženy rozpoznávači izolovaných slov. Test navíc proběhl pro dva různě rozsáhlé slovníky – jeden slovník obsahoval přesně oněch 100 testovaných slov, druhý byl podstatně větší – kolem 400 000 slov. Výsledky testu lze nejlépe ukázat tabulkou. Vzhledem k tomu, že ozvěna nebyla přidána uměle, nebylo ani možné ověřit rozpoznávání „čistých“ slov (slov bez ozvěny). Tabulka ukazuje úspěšnost rozpoznávače v procentech (1 slovo $\sim$ 1 %).

akce provedená s nahrávkami	**malý slovník**	**velký slovník**
nic	4	0
potlačen šum	2	0
potlačena ozvěna	32	1
potlačen šum, poté ozvěna	18	0
potlačen šum, ozvěna a nový šum přidán	82	19

Tabulka 1: Úspěšnost rozpoznávače pro různé nahrávky (test č. 1)

Z tabulky vidíme, že s narušenými nahrávkami si rozpoznávač vůbec neporadil – pouze 4 % i pro malý slovník. Úspěšnost se podstatně zlepšila po potlačení ozvěny – 32 %. Další zlepšení mělo logicky následovat po potlačení šumu před vlastním potlačením ozvěny. Poněkud nečekaně úspěšnost klesla na 18 %. Z toho je patrné, že rozpoznávač byl přeci jen trénován z nahrávek obsahujících třeba i nepatrný šum a proto se s nahrávkami bez šumu nedokázal dobře vyrovnat. Tento důležitý poznatek vedl k dalšímu zlepšení...

Z nahrávek byl nejprve odstraněn šum, což byl základní předpoklad pro dobré potlačení ozvěny. Následně byla ozvěna potlačena a speciálně pro rozpoznávač byl nový *bílý šum* přidán. Jeho optimální velikost jsme experimentálně určili okolo 0.5 % z celkového rozsahu[4]. Jak ukazuje poslední řádek tabulky, **úspěšnost se podstatně zlepšila**. Za zmínku také stojí, že čísla v posledním řádku nejsou stálá, neboť k nahrávkám přidáváme náhodný signál. Úspěšnost se tak může ještě změnit o cca ±3 % (pro malý slovník).

Dále byl proveden druhý podobný test. Tentokrát jsme ovšem ozvěnu přidávali uměle a tak bylo možné ověřit úspěšnost rozpoznávače i pro nahrávky bez ozvěny (poslední řádek tabulky). Dlužno podotknout, že v tomto případě bylo v nahrávkách, ve srovnání s předchozími, více šumu a méně ozvěny. Výsledky jsou ovšem podobné, alespoň co se poklesu a nárůstu úspěšnosti, vzhledem k provedené akci, týče.

akce provedená s nahrávkami	**malý slovník**	**velký slovník**
nic	19	2
potlačen šum	2	0
potlačena ozvěna	43	6
potlačen šum, poté ozvěna	24	1
potlačen šum, ozvěna a nový šum přidán	80	21
nahrávky bez rušení	98	64

Tabulka 2: Úspěšnost rozpoznávače pro různé nahrávky (test č. 2)

V posledním řádku vidíme ideál, ke kterému bychom se rádi dostali. Zatímco v případě malého slovníku jsme tomuto ideálu poměrně blízko, pro velký slovník je stále na čem pracovat.

[4]předpokládáme-li možný rozsah signálu od -1 do 1, je šum v intervalu od -0.005 do 0.005

8 Závěr

Nyní zbývá nelehký úkol, kterým je celou, téměř roční, práci zhodnotit. Začnu tedy od začátku. Hlavní zaměření diplomové práce bylo jasné – pochopit a eliminovat dva typy rušení – šum a ozvěnu. Tato rušení (hlavně ozvěna) působí nepříjemné problémy systémům komunikujícím s člověkem a možnosti těchto dialogových systémů jsou tak velmi zásadně omezeny. Cílem tak bylo „pomoci“ v tomto směru dialogovému systému InfoCity běžícímu na TUL. Tento proces zatím pokračuje a diplomová práce dle mého názoru poskytla velmi dobré předpoklady k jeho zdárnému naplnění.

Zkusím se nyní řešeným problémům věnovat trochu podrobněji. V případě šumu byla situace ve srovnání s ozvěnou o něco příznivější. Veškeré poznatky se mi podařilo realizovat a výsledný systém je velmi dobře funkční a spolehlivý. Je rovněž velmi rychlý. Hlavním problémem bylo, aby systém potlačoval skutečně pouze šum a nijak nenarušoval užitečnou informaci v signálu obsaženou. Toho se podařilo docílit s využitím optimalizačních metod. Problém šumu lze tudíž považovat za vyřešený.

Systém pro potlačení ozvěny již takto vychválit nelze. Ozvěna totiž představuje rušení ve své podstatě daleko komplikovanější. V průběhu práce jsem také velmi postrádal postřehy a hlavně výsledky jiných, kteří se ozvěnou zabývali. Nebylo totiž vůbec zřejmé, jak daleko je současný vývoj, kam se lze při potlačení ozvěny dostat a jaké jsou hranice. Výsledek proto nelze s ničím porovnat, lze „pouze“ ověřit jeho přínos pro dialogový systém. Ten je sice velmi znatelný (viz. předchozí kapitola), stále ale není zřejmé, je-li dostatečný pro reálné nasazení. Jistě by proto bylo vhodné na systému dále pracovat (vyzkoušet i jiné přístupy) a výsledky opět o něco málo či více posunout. Určitým problémem je také rychlost systému, neboť ten je založen na velmi náročném adaptivním algoritmu AP (Affine Projection). V současné době, vzhledem k rychlosti počítačů, možná již nejde o veliký handicap, nicméně možnost urychlení existuje a je poměrně škoda, že se toto v rámci práce nepodařilo udělat. Opět ale musím říci, že již nejde o elementární problém a tak se mi v této souvislosti ani nepodařilo sehnat potřebnou literaturu. Abych to ale uzavřel – systém pro potlačení ozvěny se mi podařilo realizovat, je funkční a docela jistě představuje podstatné zlepšení jak z hlediska pasivního poslechu, tak z hlediska rozpoznávače. Prostor pro další práci tu ovšem stále je.

Kromě rušení jsem se v diplomové práci dále věnoval principům a metodám rozpoznávání řeči. V této souvislosti jsem se omezil na rozpoznávání izolovaných slov, neboť vzhledem k hlavnímu zaměření práce, tj. rušení, by nebylo účelné zkoumat podstatně složitější problematiku rozpoznávání řeči spojité. Výsledkem (spíše pro ověření či vyzkoušení) je plně funkční klasifikátor izolovaných slov založený na markovských modelech.

V souvislosti s rozpoznáváním řeči lze také asi nejvíce ocenit přínos vyvinutého softwaru. Vhodnou kombinací obou systémů pro potlačení šumu a ozvěny se podařilo velmi významně přispět k úspěšnosti rozpoznávače, jehož výsledky jsou jinak, vzhledem k velkému rušení, zcela nedostatečné.

Literatura

[1] Steven L. Gay – Jacob Benesty. *Acoustic Signal Processing for Telecommunication*. Kluwer Academic Publishers, 2000. ISBN 0-7923-7814-8.

[2] Christina Breining – Pia Dreiseitel – Eberhard Hansler. Acoustic Echo Control. *IEEE Signal Processing magazine*, Vol. 16, No. 4, July 1999.

[3] Jaroslav Svoboda a kolektiv. *Telekomunikační technika, Díl 3. Telekomunikační sítě a služby*. Nakladatelství Sdělovací technika, 1999. 1. vydání. ISBN 80-901936-7-6.

[4] Jaroslav Svoboda a kolektiv. *Telekomunikační technika, Díl 1. Zprávy, signály, přenosová prostředí*. Nakladatelství Sdělovací technika, 2000. 2. vydání. ISBN 80-901936-3-3.

[5] John R. Deller – John G. Proakis – John H. L. Hansen. *Discrete-Time Processing of Speech Signals*. Macmillan Publishing Company, 1993. ISBN 0-02-328301-7.

[6] Jan Nouza. *Počítačové zpracování řeči*. TUL Liberec, 2001. ISBN 80-7083-551-6.

[7] R. E. Crochiere – L. R. Rabiner. *Multirate Digital Signal Processing*. Prentice-Hall, 1983.

[8] Jan Štecha. *Optimální rozhodování a řízení – Přednášky*. Vydavatelství ČVUT, 2002. Skripta. ISBN 80-01-02083-5.

[9] George-Othon Glentis – Kostas Berberidis – Sergios Theodoridis. Efficient Least Squares Adaptive Algoritms for FIR Transversal Filtering. *IEEE Signal Processing magazine*, Vol. 16, No. 4, July 1999.